VEL TECH HIGH TECH
Dr. RANGARAJAN Dr.SAKUNTHALA ENGINEERING COLLEGE
(An Autonomous Institution)
(An ISO 9001-2008 CERTIFIED INSTITUTION)

Approved by AICTE New Delhi and Affliated to Anna University)
No.62 Avadi -- Veltech Road Chennai 600 062.

21HC57P-NETWORKS LABORATORY

NAME :

REGISTER NO :

ROLL NO :

BRANCH :

YEAR :

SEMESTER :

VEL TECH HIGH TECH
Dr. RANGARAJAN Dr.SAKUNTHALA ENGINEERING COLLEGE
(An Autonomous Institution)
(An ISO 9001-2008 CERTIFIED INSTITUTION)
Approved by AICTE New Delhi and Affliated to Anna Universi
No.62 Avadi -- Veltech Road Chennai 600 062

BONAFIDE CERTIFICATE

NAME: ....

YEAR: .... SEMESTER: .... BRANCH: B.E COMPUTER SCIENCE &ENGINEERING

UNIVERSITY REGISTER NO: VH NO:

Certified that this is the bonafide record of work done by the above student in NETWORKS LABORATORY – 21HC57P during the academic year 2025 – 2026.

Signature of Head of the Department Signature of Lab Incharge

Submitted for the University Practical Examination held on ______________________________

At VEL TECH HIGH TECH Dr.RANGARAJAN Dr.SAKUNTHALA ENGINEERING COLLEGE,
No.60, AVADI – ALAMATHI ROAD , AVADI , CHENNAI -600 062.

Signature of Examiners :

Internal: .... External: ....

DATE: ....

VEL TECH HIGH TECH
Dr. RANGARAJAN Dr.SAKUNTHALA ENGINEERING COLLEGE

VISION:

Pursuit of excellence in technical education to create civic responsibility with competency.

MISSION:

1. To impart the attributes of global engineers to face industrial challenges with social relevance.
2. To indoctrinate as front runners through moral practices.
3. To attain the skills through lifelong learning.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING COLLEGE

VISION:

To blend academic learning process and innovative ideas producing self-confident graduates with skills and knowledge to compete in the changing world.

MISSION:

- To provide strong basic knowledge in Computer Science and Engineering.
- To motivate the students to function as a team from their innovative ideas in collaboration with the industries.
- To enable the students to lead and adapt to the growing environment.

NAME: REG NO:

PROGRAMME EDUCATIONAL OBJECTIVES

PEO-1:

Embark upon successful professional practice in Computer Science and Engineering, displaying supportive and leadership roles.

PEO-2:

Engage in professional projects requiring teamwork and making valuable contributions to design, development, and production in the practice of Computer Science and Engineering or application areas.

PEO-3:

Equip to adapt and grow with changes in technology and globalization, and to pursue higher studies and research activities.

PEO- 4:

Be capable of productive employment in the field of Computer Science and Engineering with competing technical expertise, good interpersonal skill.

PEO-5:

Utilize their broad educational experience, ethics, and professionalism to make a positive impact on their local and professional communities

PROGRAMME OUTCOMES (POs):

PO1: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2: Problem analysis: Identify, formulate, research literature, and analyze complex engineering Problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3: Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmentalconsiderations.

PO4: Conduct investigations of complex problems: Use research-based knowledge and research Methods including design of experiments, analysis and interpretation of data, and synthesis of the Information to provide valid conclusions.

PO5: Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO6: The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7: Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainabledevelopment.

PO8: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9: Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10: Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clearinstructions.

PO11: Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinaryenvironments.

PO12: Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

NAME: REG NO:

PROGRAM SPECIFIC OUTCOMES (PSOS)

By the time of graduation, the undergraduate Computer Science and Engineering students will have the ability of

- Designing Computer/Electronic based components which would serve social environment.

- Applying the current and gained knowledge and modern techniques not only in the Computers but in all related fields.

COURSE OBJECTIVES:

- To learn and use network commands.
- To learn socket programming.
- To implement and analyze various network protocols.
- To learn and use simulation tools.
- To use simulation tools to analyze the performance of various network protocols

COURCE OUTCOME:

- Implement various protocols using TCP and UDP.
- Compare the performance of different transport layer protocols.
- Use simulation tools to analyze the performance of various network protocols.
- Analyze various routing algorithms.
- Implement error correction codes.

HARDWARE:

1. Standalone desktops	30 Nos

SOFTWARE:

1. C / C++ / Java / Python / Equivalent Compiler	30Nos
2. Network simulator like NS2/Glomosim/OPNET/ Packet Tracer / Equivalent	

JAVA

Java is a high-level programming language originally developed by Sun Microsystems. Java runs on a variety of platforms, such as Windows, Mac OS, and the various versions of UNIX. Java programming were "Simple, Robust, Portable,Platform-independent, Secured, High Performance, Multithreaded, Architecture Neutral, Object-Oriented, Interpreted, and Dynamic".

NS2

NS2 stands for Network Simulator Version 2. It is an open-source event-driven simulator designed specifically for research in computer communication networks. It provides substantial support to simulate bunch of protocols like TCP, FTP, UDP, https and DSR.It simulates wired and wireless network. It is primarily Unix Based. It uses TCL as its scripting language

CO-PO MATRIX

CO	**PO1**	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO10**	**PO11**	**PO12**
CS8581.1	3		2	-	-	-	-	-	-	-	-	-
CS8581.2	3	1	2	-	-	-	-	-	-	-	-	-
CS8581.3				-	2	-	-	-	-	-	-	-
CS8581.4	2	1	2	-	-	-	-	-	-	-	-	-
CS8581.5	3	1	1	-	2	-	-	-	-	-	-	-
CS8581	3	1	2	-	2	-	-	-	-	-	-	-

1 –Low 2 – Medium 3 – High

CO-PSO MATRIX

COURSE	**PSO 3**
CS8581.1	**2**
CS8581.2	**3**
CS8581.3	**3**
CS8581.4	**3**
CS8581.5	**2**
CS8581	**3**

LIST OF EXPERIMENTS

1. Learn to use commands like tcpdump, netstat, ifconfig, nslookup and traceroute.
2. Capture ping and traceroute PDUs using a network protocol analyzer and examine.
3. Write a HTTP web client program to download a web page using TCP Sockets.
4. Applications using TCP sockets like:
    a. Echo client and echo server
    b. Chat
    c. File Transfer
5. Simulation of DNS using UDP sockets.
6. Write a code simulating ARP /RARP protocols
7. Traffic Analysis using Wireshark.
8. Study of Network simulator (NS) and Simulation of Congestion Control Algorithms using NS.
9. Study of TCP/UDP performance using Simulation tool.
10. Simulation of Distance Vector/ Link State Routing algorithm.
11. Performance evaluation of Routing protocols using Simulation tool.
12. Simulation of error correction code (like CRC).
13. Simulation of RIPandOSPFusingNS2/Qualnet/JSim/OmneT++.

NAME: REG NO:

S.NO	DATE	EXPERIMENT NAME	PAGE. NO	MARK	SIGN	
1		Learn to use commands like tcpdump, netstat, ifconfig, nslookup and traceroute.				
2		Capture ping and traceroute PDUs using a network protocol analyzer and examine.				
3		Write a HTTP web client program to download a web page using TCP Sockets.				
4		Applications using TCP sockets like: a.Echo client and echo server b.Chat c.File Transfer				
5		Simulation of DNS using UDP sockets.				
6		Write a code simulating ARP /RARP protocols				
7		Traffic Analysis using Wireshark.				
8		Study of Network simulator (NS) and Simulation of Congestion Control Algorithms using NS.				
9		Study of TCP/UDP performance using Simulation tool.				
10		a. Simulation of Distance Vector. b. Link State Routing algorithm.				
11		Performance evaluation of Routing protocols using Simulation tool.				
12		Simulation of error correction code (like CRC).				
13		Simulation of RIPandOSPFusingNS2/Qualnet/JSim/OmneT++.				
	TOPIC BEYOND SYLLABUS					
14.		a. Simulation of Go Back N protocol b. Carrier Sense Multiple Access.				

NAME: REG NO:

EX.NO :1 & 2	**Learn to use commands like tcpdump, netstat, ifconfig, nslookup trace route .Capture ping and traceroute PDUs using a network protocol analy and examine.**
DATE:	

AIM:

PROGRAM:

nslookup
nslookup-> gives corresponding ipaddress of the domain name.
Ipconfig(windows)/ifconfig(Linux)->displays ipaddress

Ifconfig(Linux)->displays ipaddress
Ping
Ping->command 2 check computers are reachable
Tracert
Tracert->traces the path of the intermediate devices it goes through to reach destination
Netstat
Netstat(network statistics)->display current network connections and port activity
Localaddress->our computer ipaddress:our port

Foreign address->remote computer ipaddress:remote port(for websites shows protocol)

Netstat -n->shows numbers not domain names
Netstat -a->displays active connections and which tcp and udp ports are listening for a connection.
Netstat -b->name of the program which makes these connections
Netstat -f->fully qualified domain name
Netstat -e->ethernet statistics
Netstat -r->shows routing table.
Arp
Arp(address resolution protocol)->resolve ipaddresses to mac-addresses
arp -a(shows connects hosts ipaddress and macaddress)

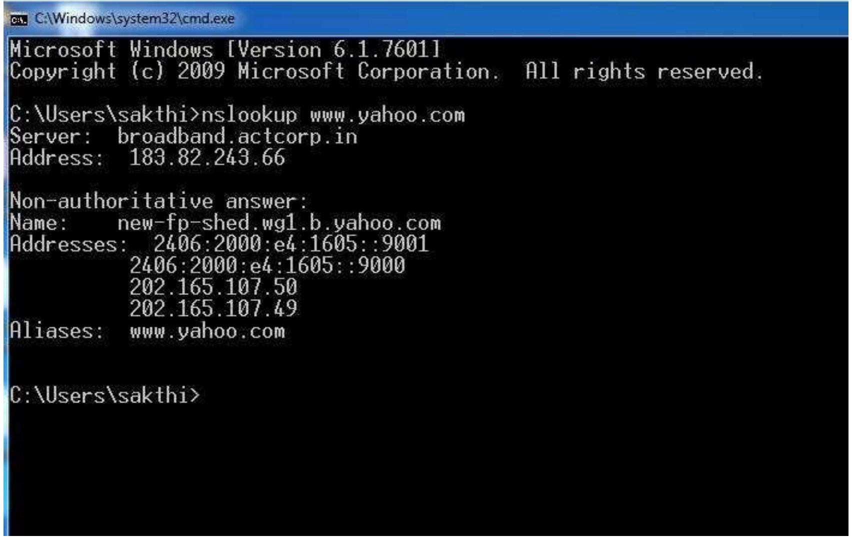
Tcpdump
Tcpdump->CLI(command line interface)utility that allows you to capture and analyze network traffic.
1) which tcpdump->to check tcpdump installed or not
2) sudo tcpdump -D->displays which interfaces are available for capture.
3) sudo tcpdump -I any
4) sudo tcpdump -I any -c 5(limiting to 5 captured packets)

NAME: REG NO:

5) sudo tcpdump -I any -c 5 -nn(don’t show domain and port as names)
6) sudo tcpdump -I any -c 5 icmp(filtering icmp packets)==ping google.com in another terminal
7) sudo tcpdump -I any -c 5 host 127.0.0.0.1 =>ping google.com from another terminal
8) sudo tcpdump -I any -c 5 port 80(webserver)->open google.com

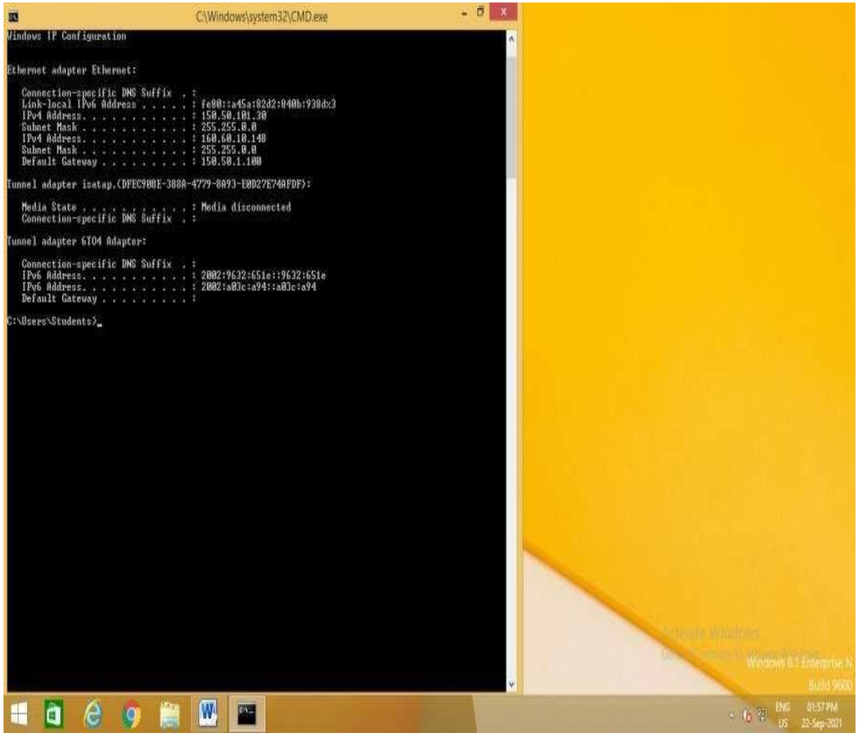
OUTPUT:

nslookup



```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\sakthi>nslookup www.yahoo.com
Server:  broadband.actcorp.in
Address:  183.82.243.66

Non-authoritative answer:
Name:    new-fp-shed.wg1.b.yahoo.com
Addresses:  2406:2000:e4:1605::9001
          2406:2000:e4:1605::9000
          202.165.107.50
          202.165.107.49
Aliases:  www.yahoo.com


C:\Users\sakthi>
```

Ipconfig(windows)/ifconfig(Linux)->displays ipaddress

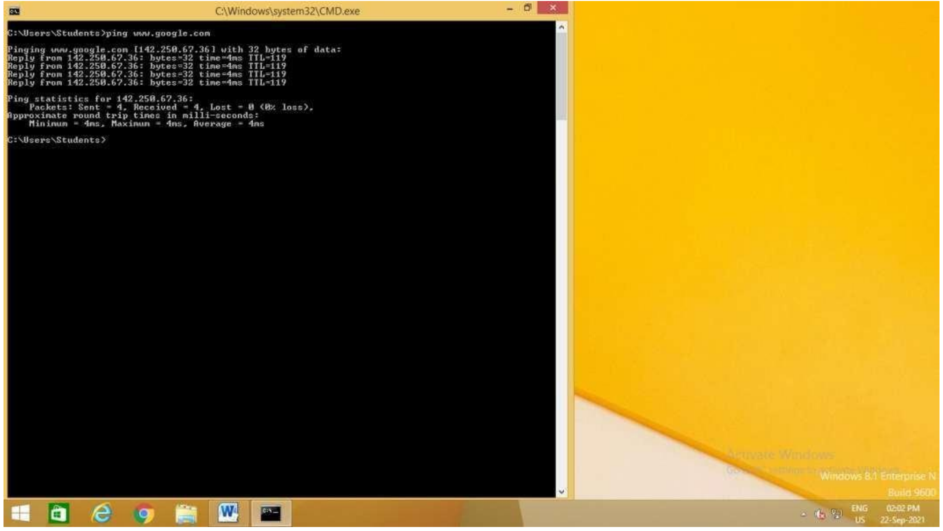


```
Windows IP Configuration


Ethernet adapter Ethernet:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::a45a:82d2:840b:938d%3
   IPv4 Address. . . . . . . . . . . : 150.50.101.30
   Subnet Mask . . . . . . . . . . . : 255.255.0.0
   IPv4 Address. . . . . . . . . . . : 160.60.10.148
   Subnet Mask . . . . . . . . . . . : 255.255.0.0
   Default Gateway . . . . . . . . . : 150.50.1.100

Tunnel adapter isatap.{DFEC908E-388A-4779-8A93-E0D27E74AFDF}:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Tunnel adapter 6TO4 Adapter:

   Connection-specific DNS Suffix  . :
   IPv6 Address. . . . . . . . . . . : 2002:9632:651e::9632:651e
   IPv6 Address. . . . . . . . . . . : 2002:a03c:a94::a03c:a94
   Default Gateway . . . . . . . . . :

C:\Users\Students>_
```

ifconfig(Linux)->displays ipaddress

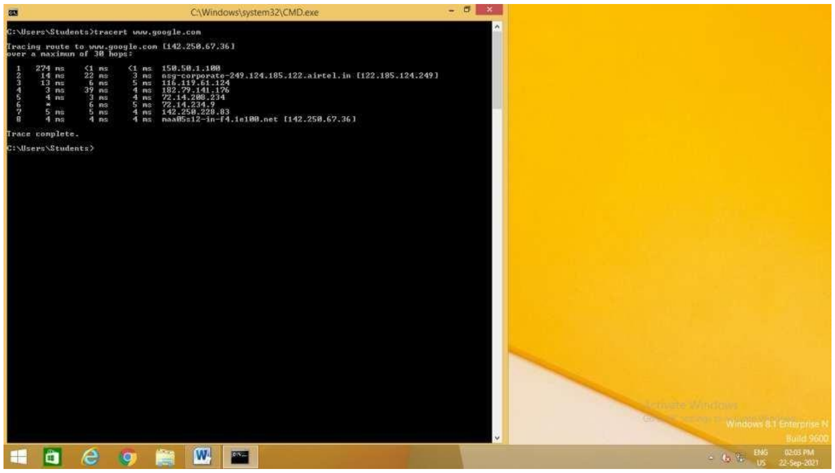
ping

ping->command 2 check computers are reachable



```
C:\Users\Students>ping www.google.com

Pinging www.google.com [142.250.67.36] with 32 bytes of data:
Reply from 142.250.67.36: bytes=32 time=4ms TTL=119
Reply from 142.250.67.36: bytes=32 time=4ms TTL=119
Reply from 142.250.67.36: bytes=32 time=4ms TTL=119
Reply from 142.250.67.36: bytes=32 time=4ms TTL=119

Ping statistics for 142.250.67.36:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 4ms, Maximum = 4ms, Average = 4ms

C:\Users\Students>
```

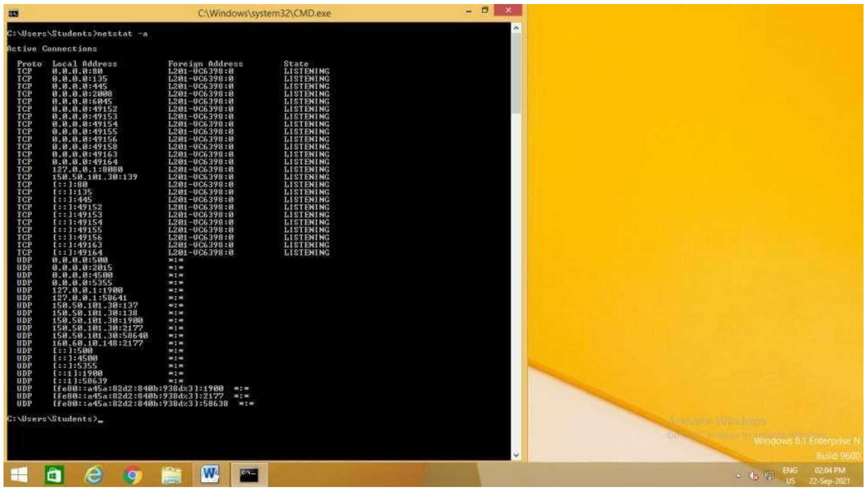
tracert



```
C:\Users\Students>tracert www.google.com

Tracing route to www.google.com [142.250.67.36]
over a maximum of 30 hops:

  1   274 ms    <1 ms    <1 ms  150.50.1.100
  2    14 ms    22 ms     3 ms  asg-corporate-249.124.185.122.airtel.in [122.185.124.249]
  3    13 ms     6 ms     5 ms  116.119.61.124
  4     3 ms    39 ms     4 ms  182.79.141.176
  5     4 ms     3 ms     4 ms  72.14.208.234
  6     *        6 ms     5 ms  72.14.234.9
  7     5 ms     5 ms     4 ms  142.250.228.83
  8     4 ms     4 ms     4 ms  maa05s12-in-f4.1e100.net [142.250.67.36]

Trace complete.

C:\Users\Students>
```

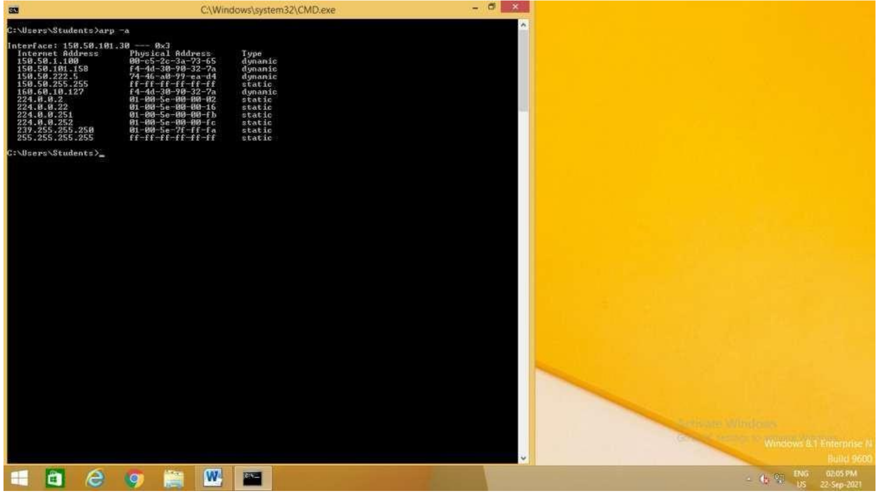
NAME: REG NO:

netstat



```
C:\Users\Students>netstat -a

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    0.0.0.0:80             L201-VC6398:0          LISTENING
  TCP    0.0.0.0:135            L201-VC6398:0          LISTENING
  TCP    0.0.0.0:445            L201-VC6398:0          LISTENING
  TCP    0.0.0.0:2008           L201-VC6398:0          LISTENING
  TCP    0.0.0.0:6045           L201-VC6398:0          LISTENING
  TCP    0.0.0.0:49152          L201-VC6398:0          LISTENING
  TCP    0.0.0.0:49153          L201-VC6398:0          LISTENING
  TCP    0.0.0.0:49154          L201-VC6398:0          LISTENING
  TCP    0.0.0.0:49155          L201-VC6398:0          LISTENING
  TCP    0.0.0.0:49156          L201-VC6398:0          LISTENING
  TCP    0.0.0.0:49158          L201-VC6398:0          LISTENING
  TCP    0.0.0.0:49163          L201-VC6398:0          LISTENING
  TCP    0.0.0.0:49164          L201-VC6398:0          LISTENING
  TCP    127.0.0.1:8080         L201-VC6398:0          LISTENING
  TCP    150.50.101.30:139      L201-VC6398:0          LISTENING
  TCP    [::]:80                L201-VC6398:0          LISTENING
  TCP    [::]:135               L201-VC6398:0          LISTENING
  TCP    [::]:445               L201-VC6398:0          LISTENING
  TCP    [::]:49152             L201-VC6398:0          LISTENING
  TCP    [::]:49153             L201-VC6398:0          LISTENING
  TCP    [::]:49154             L201-VC6398:0          LISTENING
  TCP    [::]:49155             L201-VC6398:0          LISTENING
  TCP    [::]:49156             L201-VC6398:0          LISTENING
  TCP    [::]:49163             L201-VC6398:0          LISTENING
  TCP    [::]:49164             L201-VC6398:0          LISTENING
  UDP    0.0.0.0:500            *:*
  UDP    0.0.0.0:2015           *:*
  UDP    0.0.0.0:4500           *:*
  UDP    0.0.0.0:5355           *:*
  UDP    127.0.0.1:1900         *:*
  UDP    127.0.0.1:58641        *:*
  UDP    150.50.101.30:137      *:*
  UDP    150.50.101.30:138      *:*
  UDP    150.50.101.30:1900     *:*
  UDP    150.50.101.30:2177     *:*
  UDP    150.50.101.30:58640    *:*
  UDP    160.60.10.148:2177     *:*
  UDP    [::]:500               *:*
  UDP    [::]:4500              *:*
  UDP    [::]:5355              *:*
  UDP    [::1]:1900             *:*
  UDP    [::1]:58639            *:*
  UDP    [fe80::a45a:82d2:840b:938d%3]:1900  *:*
  UDP    [fe80::a45a:82d2:840b:938d%3]:2177  *:*
  UDP    [fe80::a45a:82d2:840b:938d%3]:58638  *:*

C:\Users\Students>_
```

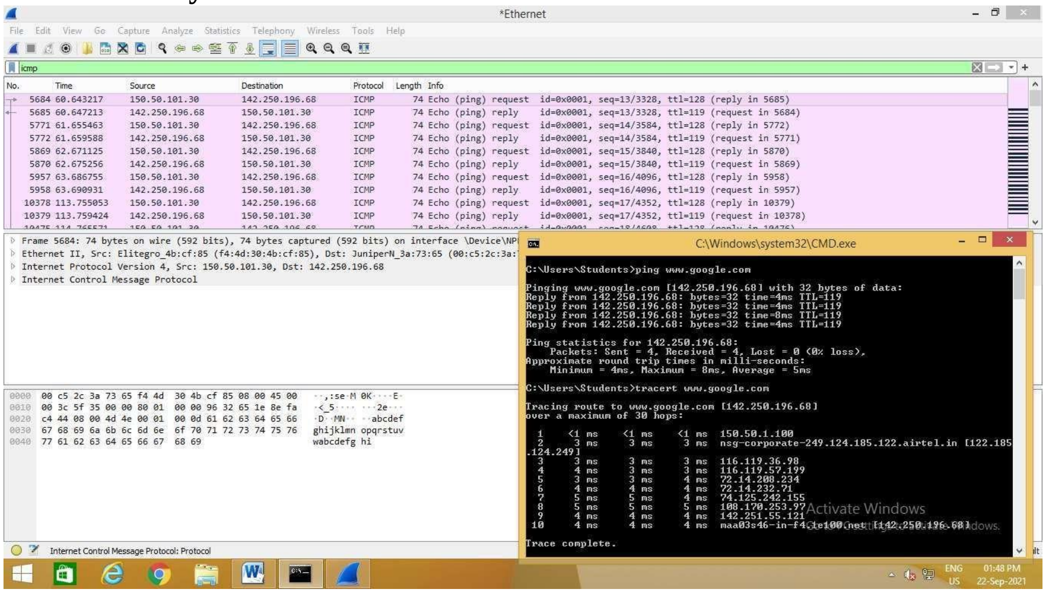
arp



```
C:\Users\Students>arp -a

Interface: 150.50.101.30 --- 0x3
  Internet Address      Physical Address      Type
  150.50.1.100          00-c5-2c-3a-73-65     dynamic
  150.50.101.158        f4-4d-30-90-32-7a     dynamic
  150.50.222.5          74-46-a0-99-ea-d4     dynamic
  150.50.255.255        ff-ff-ff-ff-ff-ff     static
  160.60.10.127         f4-4d-30-90-32-7a     dynamic
  224.0.0.2             01-00-5e-00-00-02     static
  224.0.0.22            01-00-5e-00-00-16     static
  224.0.0.251           01-00-5e-00-00-fb     static
  224.0.0.252           01-00-5e-00-00-fc     static
  239.255.255.250       01-00-5e-7f-ff-fa     static
  255.255.255.255       ff-ff-ff-ff-ff-ff     static

C:\Users\Students>_
```

NAME: REG NO:

Network analyzer: wireshark


*Ethernet
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
icmp
C:\Windows\system32\CMD.exe
Activate Windows
ENG US
01:48 PM
22-Sep-2021

```
No.     Time        Source          Destination     Protocol Length Info
   5684 60.643217   150.50.101.30   142.250.196.68  ICMP     74 Echo (ping) request  id=0x0001, seq=13/3328, ttl=128 (reply in 5685)
   5685 60.647213   142.250.196.68  150.50.101.30   ICMP     74 Echo (ping) reply    id=0x0001, seq=13/3328, ttl=119 (request in 5684)
   5771 61.655463   150.50.101.30   142.250.196.68  ICMP     74 Echo (ping) request  id=0x0001, seq=14/3584, ttl=128 (reply in 5772)
   5772 61.659588   142.250.196.68  150.50.101.30   ICMP     74 Echo (ping) reply    id=0x0001, seq=14/3584, ttl=119 (request in 5771)
   5869 62.671125   150.50.101.30   142.250.196.68  ICMP     74 Echo (ping) request  id=0x0001, seq=15/3840, ttl=128 (reply in 5870)
   5870 62.675256   142.250.196.68  150.50.101.30   ICMP     74 Echo (ping) reply    id=0x0001, seq=15/3840, ttl=119 (request in 5869)
   5957 63.686755   150.50.101.30   142.250.196.68  ICMP     74 Echo (ping) request  id=0x0001, seq=16/4096, ttl=128 (reply in 5958)
   5958 63.690931   142.250.196.68  150.50.101.30   ICMP     74 Echo (ping) reply    id=0x0001, seq=16/4096, ttl=119 (request in 5957)
  10378 113.755053  150.50.101.30   142.250.196.68  ICMP     74 Echo (ping) request  id=0x0001, seq=17/4352, ttl=128 (reply in 10379)
  10379 113.759424  142.250.196.68  150.50.101.30   ICMP     74 Echo (ping) reply    id=0x0001, seq=17/4352, ttl=119 (request in 10378)
```

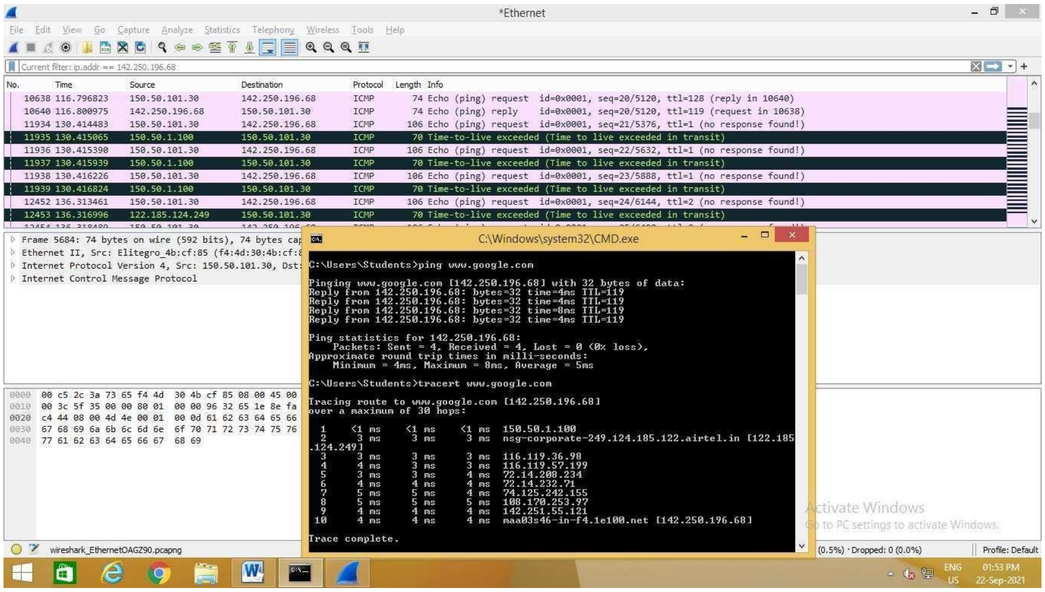
```
▷ Frame 5684: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface \Device\NP
▷ Ethernet II, Src: Elitegro_4b:cf:85 (f4:4d:30:4b:cf:85), Dst: JuniperN_3a:73:65 (00:c5:2c:3a:
▷ Internet Protocol Version 4, Src: 150.50.101.30, Dst: 142.250.196.68
▷ Internet Control Message Protocol
```

```
0000  00 c5 2c 3a 73 65 f4 4d  30 4b cf 85 08 00 45 00   ··,:se·M 0K····E·
0010  00 3c 5f 35 00 00 80 01  00 00 96 32 65 1e 8e fa   ·<_5···· ···2e···
0020  c4 44 08 00 4d 4e 00 01  00 0d 61 62 63 64 65 66   ·D··MN·· ··abcdef
0030  67 68 69 6a 6b 6c 6d 6e  6f 70 71 72 73 74 75 76   ghijklmn opqrstuv
0040  77 61 62 63 64 65 66 67  68 69                     wabcdefg hi
```

Internet Control Message Protocol: Protocol

```
C:\Users\Students>ping www.google.com

Pinging www.google.com [142.250.196.68] with 32 bytes of data:
Reply from 142.250.196.68: bytes=32 time=4ms TTL=119
Reply from 142.250.196.68: bytes=32 time=4ms TTL=119
Reply from 142.250.196.68: bytes=32 time=8ms TTL=119
Reply from 142.250.196.68: bytes=32 time=4ms TTL=119

Ping statistics for 142.250.196.68:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 4ms, Maximum = 8ms, Average = 5ms

C:\Users\Students>tracert www.google.com

Tracing route to www.google.com [142.250.196.68]
over a maximum of 30 hops:

  1    <1 ms    <1 ms    <1 ms  150.50.1.100
  2     3 ms     3 ms     3 ms  nsg-corporate-249.124.185.122.airtel.in [122.185
.124.249]
  3     3 ms     3 ms     3 ms  116.119.36.98
  4     4 ms     3 ms     3 ms  116.119.57.199
  5     3 ms     3 ms     4 ms  72.14.208.234
  6     4 ms     4 ms     4 ms  72.14.232.71
  7     5 ms     5 ms     4 ms  74.125.242.155
  8     5 ms     5 ms     5 ms  108.170.253.97
  9     4 ms     4 ms     4 ms  142.251.55.121
 10     4 ms     4 ms     4 ms  maa03s46-in-f4.1e100.net [142.250.196.68]

Trace complete.
```

Captured: ping and traceroute PDU


*Ethernet
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
C:\Windows\system32\CMD.exe
Activate Windows
Go to PC settings to activate Windows.
ENG US
01:53 PM
22-Sep-2021

Current filter: ip.addr == 142.250.196.68

```
No.     Time        Source           Destination     Protocol Length Info
  10638 116.796823  150.50.101.30    142.250.196.68  ICMP     74 Echo (ping) request  id=0x0001, seq=20/5120, ttl=128 (reply in 10640)
  10640 116.800975  142.250.196.68   150.50.101.30   ICMP     74 Echo (ping) reply    id=0x0001, seq=20/5120, ttl=119 (request in 10638)
  11934 130.414483  150.50.101.30    142.250.196.68  ICMP    106 Echo (ping) request  id=0x0001, seq=21/5376, ttl=1 (no response found!)
  11935 130.415065  150.50.1.100     150.50.101.30   ICMP     70 Time-to-live exceeded (Time to live exceeded in transit)
  11936 130.415390  150.50.101.30    142.250.196.68  ICMP    106 Echo (ping) request  id=0x0001, seq=22/5632, ttl=1 (no response found!)
  11937 130.415939  150.50.1.100     150.50.101.30   ICMP     70 Time-to-live exceeded (Time to live exceeded in transit)
  11938 130.416226  150.50.101.30    142.250.196.68  ICMP    106 Echo (ping) request  id=0x0001, seq=23/5888, ttl=1 (no response found!)
  11939 130.416824  150.50.1.100     150.50.101.30   ICMP     70 Time-to-live exceeded (Time to live exceeded in transit)
  12452 136.313461  150.50.101.30    142.250.196.68  ICMP    106 Echo (ping) request  id=0x0001, seq=24/6144, ttl=2 (no response found!)
  12453 136.316996  122.185.124.249  150.50.101.30   ICMP     70 Time-to-live exceeded (Time to live exceeded in transit)
```

```
▷ Frame 5684: 74 bytes on wire (592 bits), 74 bytes cap
▷ Ethernet II, Src: Elitegro_4b:cf:85 (f4:4d:30:4b:cf:8
▷ Internet Protocol Version 4, Src: 150.50.101.30, Dst:
▷ Internet Control Message Protocol
```

```
0000  00 c5 2c 3a 73 65 f4 4d  30 4b cf 85 08 00 45 00
0010  00 3c 5f 35 00 00 80 01  00 00 96 32 65 1e 8e fa
0020  c4 44 08 00 4d 4e 00 01  00 0d 61 62 63 64 65 66
0030  67 68 69 6a 6b 6c 6d 6e  6f 70 71 72 73 74 75 76
0040  77 61 62 63 64 65 66 67  68 69
```

wireshark_EthernetOAGZ90.pcapng (0.5%) · Dropped: 0 (0.0%) Profile: Default

```
C:\Users\Students>ping www.google.com

Pinging www.google.com [142.250.196.68] with 32 bytes of data:
Reply from 142.250.196.68: bytes=32 time=4ms TTL=119
Reply from 142.250.196.68: bytes=32 time=4ms TTL=119
Reply from 142.250.196.68: bytes=32 time=8ms TTL=119
Reply from 142.250.196.68: bytes=32 time=4ms TTL=119

Ping statistics for 142.250.196.68:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 4ms, Maximum = 8ms, Average = 5ms

C:\Users\Students>tracert www.google.com

Tracing route to www.google.com [142.250.196.68]
over a maximum of 30 hops:

  1    <1 ms    <1 ms    <1 ms  150.50.1.100
  2     3 ms     3 ms     3 ms  nsg-corporate-249.124.185.122.airtel.in [122.185
.124.249]
  3     3 ms     3 ms     3 ms  116.119.36.98
  4     4 ms     3 ms     3 ms  116.119.57.199
  5     3 ms     3 ms     4 ms  72.14.208.234
  6     4 ms     4 ms     4 ms  72.14.232.71
  7     5 ms     5 ms     4 ms  74.125.242.155
  8     5 ms     5 ms     5 ms  108.170.253.97
  9     4 ms     4 ms     4 ms  142.251.55.121
 10     4 ms     4 ms     4 ms  maa03s46-in-f4.1e100.net [142.250.196.68]

Trace complete.
```

NAME: REG NO:

RESULT:

NAME: REG NO:

Ex.No: 3	**Write a HTTP web client program to download a web page using TCP sockets**
DATE:	

AIM:

ALGORITHM:

SERVER SIDE:

1) Start the program.

2) Create a server socket to activate the port address.

3) Create a socket for the server socket which accepts the connection.

4) After establishing connection receive url from client.

5) Download the content of the url received and send the data to client.

6) Close the socket.

7) End the program

CLIENT SIDE:

1) Start the program.

2) Create a socket which binds the Ip address of server and the port address to acquire service.

3) After establishing connection send the url to server.

4) Open a file and store the received data into the file.

5) Close the socket.

6) End the program.

NAME: REG NO:

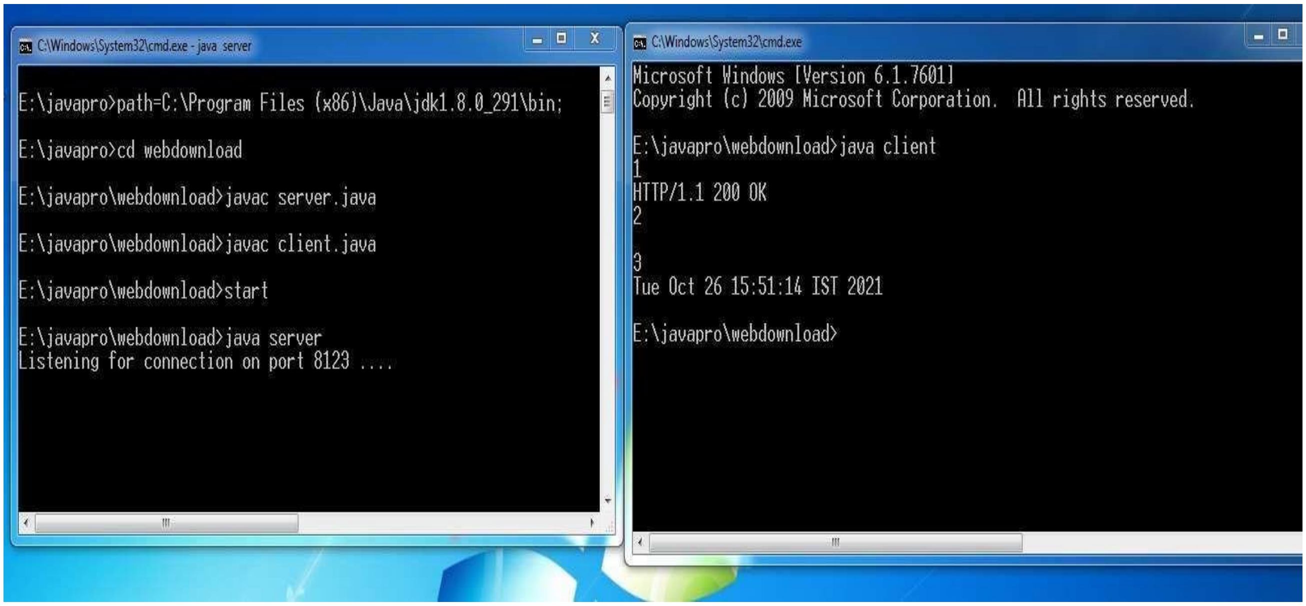
server.java

```
import java.io.IOException;

import java.net.ServerSocket;

import java.net.Socket;

import java.util.Date;

public class server {

public static void main(String args[]) throws IOException {

    ServerSocket server = new ServerSocket(8123);

System.out.println("Listening for connection on port 8123 ..... ");

while (true) {

try (Socket socket = server.accept()) {

        Date today = new Date();

        String httpResponse = "HTTP/1.1 200 OK\r\n\r\n" + today;

socket.getOutputStream()

           .write(httpResponse.getBytes("UTF-8"));

      }

    }

  }


}
```

client.java

```
import java.io.BufferedReader;

import java.io.BufferedWriter;

import java.io.InputStreamReader;

import java.io.OutputStreamWriter;

import java.io.PrintWriter;

import java.net.Socket;

public class client {

public static void main(String[] args) throws Exception {

try {

Socket socket = new Socket("localhost",8123);

PrintWriter out = new PrintWriter(new BufferedWriter(new
OutputStreamWriter(socket.getOutputStream())));

out.println("GET /index.html HTTP/1.0");

out.println();

out.flush();

BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));

String inputLine;

int count = 0;

while ((inputLine = in.readLine()) != null) {

count++;

System.out.println(count);

System.out.println(inputLine);          }                    in.close();        }

catch (Exception e) {

e.printStackTrace();          }        }        }
```

OUTPUT:


C:\Windows\System32\cmd.exe - java server
C:\Windows\System32\cmd.exe

```
E:\javapro>path=C:\Program Files (x86)\Java\jdk1.8.0_291\bin;

E:\javapro>cd webdownload

E:\javapro\webdownload>javac server.java

E:\javapro\webdownload>javac client.java

E:\javapro\webdownload>start

E:\javapro\webdownload>java server
Listening for connection on port 8123 ....
```

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

E:\javapro\webdownload>java client
1
HTTP/1.1 200 OK
2

3
Tue Oct 26 15:51:14 IST 2021

E:\javapro\webdownload>
```

RESULT:

NAME: REG NO:

Ex.No: 4	**Applications using TCP sockets like: Echo client and echo server, Chat and File Transfer**
DATE:	

AIM

3A)ECHO CLIENT AND ECHO SERVER

ALGORITHM

SERVER SIDE

1. Start the program.

2. Create a server socket to activate the port address.

3. Create a socket for the server socket which accepts the connection.

4. After establishing connection receive the data from client.

5. Print and send the same data to client.

6. Close the socket.

7. End the program.

ALGORITHM

CLIENT SIDE

1. Start the program.

2. Create a socket which binds the Ip address of server and the port address to acquire service.

3. After establishing connection send a data to server.

4. Receive and print the same data from server.

5. Close the socket.

6. End the program.

NAME: REG NO:

PROGRAM:

eclient.java:

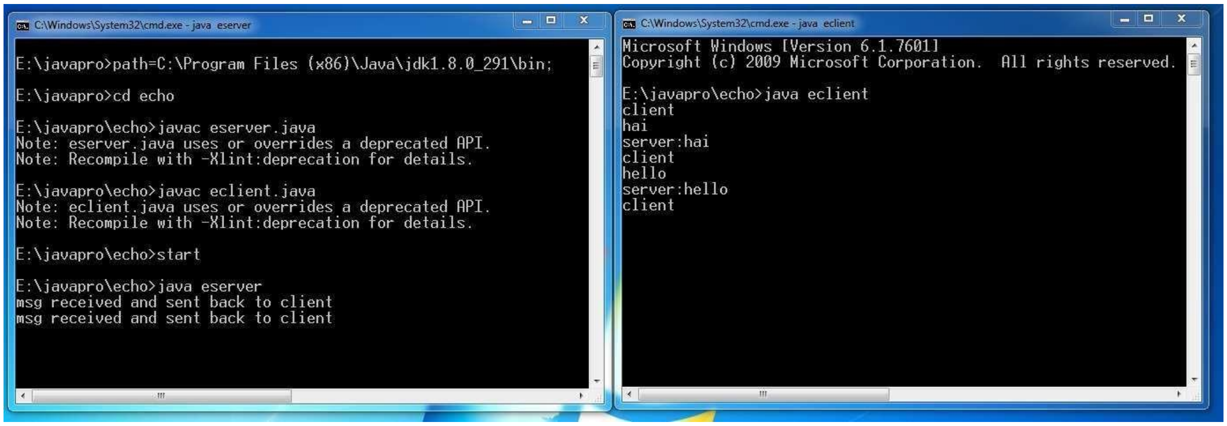
```
import java.io.*;
import java.net.*;
public class eclient
{
public static void main(String args[]){
Socket c=null;
String line;
DataInputStream is,is1;
PrintStream os;
try{
c=new Socket("localhost",8080);
}
catch(IOException e){
System.out.println(e);
}
try{
os=new PrintStream(c.getOutputStream());
is=new DataInputStream(System.in);
is1=new DataInputStream(c.getInputStream());
do{
System.out.println("client");
line=is.readLine();
os.println(line);
if(!line.equals("exit"))
System.out.println("server:"+is1.readLine());
}while(!line.equals("exit"));}
catch(IOException e){
System.out.println("socket closed");
}}}
```

NAME: REG NO:

eserver.java:

```
import java.io.*;

import java.net.*;

public class eserver

{public static void main(String args[])throws IOException

{ServerSocket s=null;

String line;

DataInputStream is;

PrintStream ps;

Socket c=null;

try{

s=new ServerSocket(8080);}

catch(IOException e){

System.out.println(e);}

try{

c=s.accept();

is=new DataInputStream(c.getInputStream());

ps=new PrintStream(c.getOutputStream());

while(true){

line=is.readLine();

System.out.println("msg received and sent back to client");

ps.println(line);}}

catch(IOException e){System.out.println(e);}}}
```

OUTPUT



```
E:\javapro>path=C:\Program Files (x86)\Java\jdk1.8.0_291\bin;

E:\javapro>cd echo

E:\javapro\echo>javac eserver.java
Note: eserver.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.

E:\javapro\echo>javac eclient.java
Note: eclient.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.

E:\javapro\echo>start

E:\javapro\echo>java eserver
msg received and sent back to client
msg received and sent back to client
```

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

E:\javapro\echo>java eclient
client
hai
server:hai
client
hello
server:hello
client
```

3B).CLIENT- SERVER APPLICATION FOR CHAT

ALGORITHM

CLIENT

1. Start the program

2. Include necessary package in java

3. To create a socket in client to server.

4. The client establishes a connection to the server.

5. The client accept the connection and to send the data from client to server.

6. The client communicates the server to send the end of the message

7. Stop the program.

SERVER

1. Start the program

2. Include necessary package in java

3. To create a socket in server to client

4. The server establishes a connection to the client.

5. The server accept the connection and to send the data from server to client and

6. vice versa

7. The server communicates the client to send the end of the message.

8. Stop the program.

PROGRAM

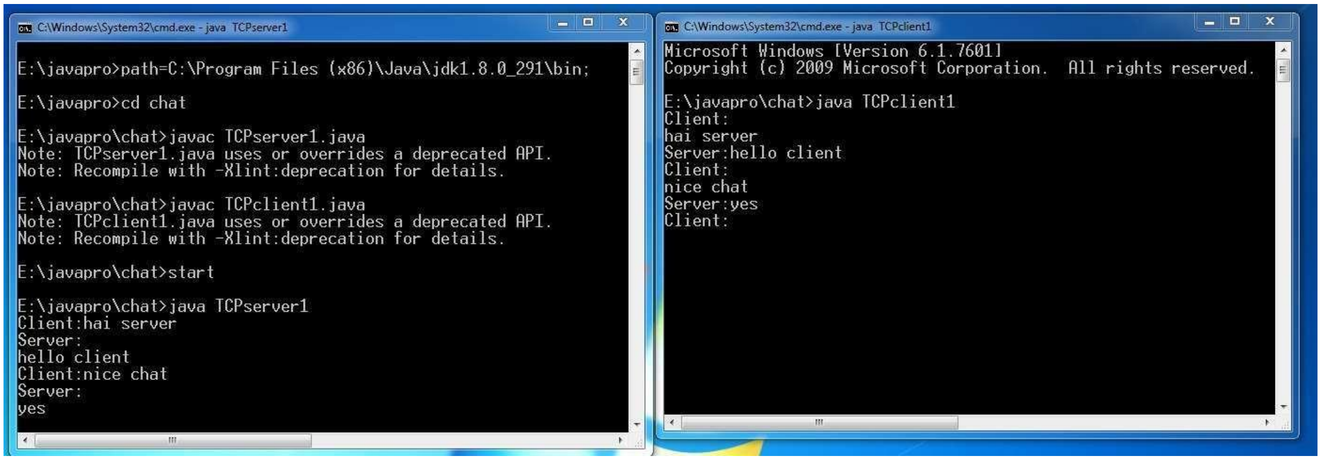
TCPserver1.java

```
import  java.net.*;
import java.io.*;
public class TCPserver1
{
public static void main(String arg[])
{
ServerSocket s=null;
String line;
DataInputStream is=null,is1=null;
PrintStream os=null;
Socket c=null;
try{
s=new ServerSocket(9999);
}
catch(IOException e){
System.out.println(e);
}
try
{
c=s.accept();
is=new DataInputStream(c.getInputStream());
is1=new DataInputStream(System.in);
os=new PrintStream(c.getOutputStream());
do

{

line=is.readLine();

System.out.println("Client:"+line);

System.out.println("Server:");

line=is1.readLine();

os.println(line);

}

while(line.equalsIgnoreCase("quit")==false);

is.close();

os.close();

}

catch(IOException e)

{

System.out.println(e);

}

}

}
```

TCPclient1.java

```
import java.net.*;

import java.io.*;

public class TCPclient1

{

public static void main(String arg[])

{

Socket c=null;

String line;

DataInputStream is,is1;

PrintStream os;

try

{

c=new Socket("localhost",9999);

}

catch(IOException e)

{

System.out.println(e);

}

try

{

os=new PrintStream(c.getOutputStream());

is=new DataInputStream(System.in);

is1=new DataInputStream(c.getInputStream());
do{
System.out.println("Client:");
line=is.readLine();
os.println(line);
System.out.println("Server:" + is1.readLine());
}
while(line.equalsIgnoreCase("quit")==false);
is1.close();
os.close();
}
catch(IOException e)
{
System.out.println("Socket Closed!Message Passing is over");
}}
```

OUTPUT:



```
E:\javapro>path=C:\Program Files (x86)\Java\jdk1.8.0_291\bin;

E:\javapro>cd chat

E:\javapro\chat>javac TCPserver1.java
Note: TCPserver1.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.

E:\javapro\chat>javac TCPclient1.java
Note: TCPclient1.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.

E:\javapro\chat>start

E:\javapro\chat>java TCPserver1
Client:hai server
Server:
hello client
Client:nice chat
Server:
yes
```

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

E:\javapro\chat>java TCPclient1
Client:
hai server
Server:hello client
Client:
nice chat
Server:yes
Client:
```

NAME: REG NO:

C) FILE TRANSFER IN CLIENT & SERVER

ALGORITHM

SERVER SIDE

1. Start.

2. Implement a server socket that listens to a particular port number.

3. Server reads the filename and sends the data stored in the file for the‘get’ request.

4. It reads the data from the input stream and writes it to a file in theserver for the ‘put’ instruction.

5. Exit upon client’s request.

6. Stop.

CLIENT SIDE

1. Start.

2. Establish a connection between the Client and Server.

3. Socket ss=new Socket(InetAddress.getLocalHost(),1100);

4. Implement a client that can send two requests.

i) To get a file from the server.

ii) To put or send a file to the server.

5. After getting approval from the server, the client either get file from the server or send

6. Send file to the server.

NAME: REG NO:

PROGRAM

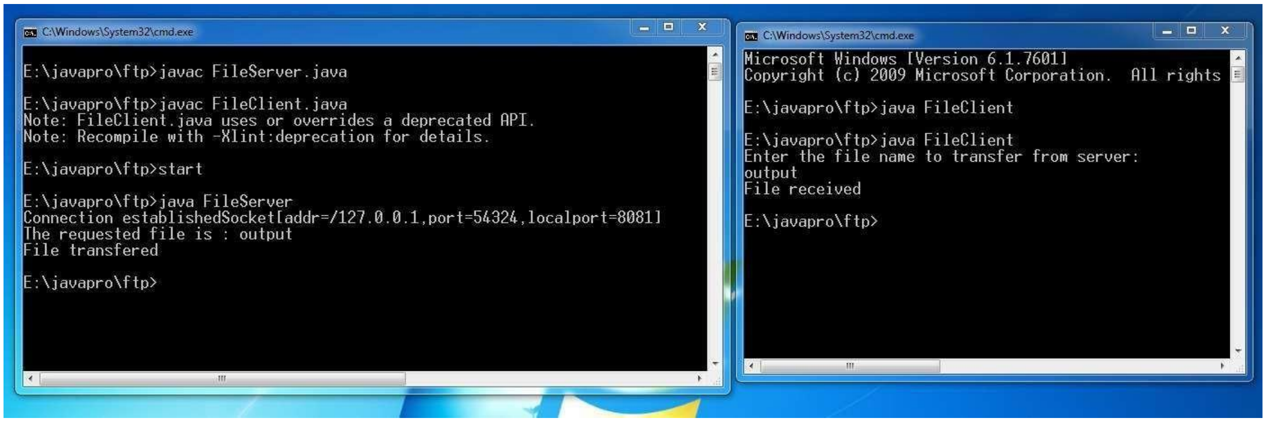
CLIENT SIDE (FileClient.java)

```
import java.net.*;
import java.io.*;
class FileClient {
    public static void main(String srgs[])throws IOException
    {
        Socket s=null;
        BufferedReader get=null;
        PrintWriter put=null;
        try
        {
            s=new Socket("127.0.0.1",8081);
            get=new BufferedReader(new InputStreamReader(s.getInputStream()));
            put=new PrintWriter(s.getOutputStream(),true);
        }
        catch(Exception e)
        {
            System.exit(0);
        }
        String u,f;
        System.out.println("Enter the file name to transfer from server:");
        DataInputStream dis=new DataInputStream(System.in);
        f=dis.readLine();
        put.println(f);
        File f1=new File("output");
        FileOutputStream fs=new FileOutputStream(f1);
        while((u=get.readLine())!=null)
        {
            byte jj[]=u.getBytes();
            fs.write(jj);
        }
        fs.close();
        System.out.println("File received");
        s.close();
    }
}
```

SERVER SIDE (FileServer.java)

```
import java.io.*;

import java.net.*;

public class FileServer {

  public static void main(String args[])throws IOException

  {

    ServerSocket ss=null;

    try        {

      ss=new ServerSocket(8081);

    }

    catch(IOException e)        {

      System.out.println("couldn't listen");

      System.exit(0);

    }

    Socket cs=null;

    try        {

      cs=ss.accept();

      System.out.println("Connection  established"+cs);

    }

    catch(Exception e)        {

      System.out.println("Accept failed");

      System.exit(1);

    }

    PrintWriter put=new PrintWriter(cs.getOutputStream(),true);

    BufferedReader st=new BufferedReader(new InputStreamReader(cs.getInputStream()));
    String s=st.readLine();
    System.out.println("The requested file is : "+s);
    File f=new File(s);
    if(f.exists())
    {
      BufferedReader d=new BufferedReader(new FileReader(s));
      String line;
      while((line=d.readLine())!=null)
      {  put.write(line);
         put.flush();           }
      d.close();
      System.out.println("File transfered");
      cs.close();
      ss.close();
    }    }  }
```

OUTPUT



```
E:\javapro\ftp>javac FileServer.java

E:\javapro\ftp>javac FileClient.java
Note: FileClient.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.

E:\javapro\ftp>start

E:\javapro\ftp>java FileServer
Connection establishedSocket[addr=/127.0.0.1,port=54324,localport=8081]
The requested file is : output
File transfered

E:\javapro\ftp>
```

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights

E:\javapro\ftp>java FileClient

E:\javapro\ftp>java FileClient
Enter the file name to transfer from server:
output
File received

E:\javapro\ftp>
```

RESULT:

NAME: REG NO:

Ex.No: 5	**Simulation of DNS using UDP Sockets.**
DATE:	

AIM

ALGORITHM

CLIENT SIDE

1. Create a datagram socket

2. Get domain name from user

3. Create a datagram packet and send domain name to the server

4. Create a datagram packet to receive server message

5. Read server's response

6. If ip address then display it else display "Domain does not exist"

7. Close the client socket

SERVER SIDE

1. Create an array of hosts and its ip address in another array

2. Create a datagram socket and bind it to a port

3. Create a datagram packet to receive client request

4. Read the domain name from client to be resolved

5. Lookup the host array for the domain name

6. If found then retrieve corresponding address

7. Create a datagram packet and send ip address to client

8. Repeat steps 3-7 to resolve further requests from clients

9. Close the server socket

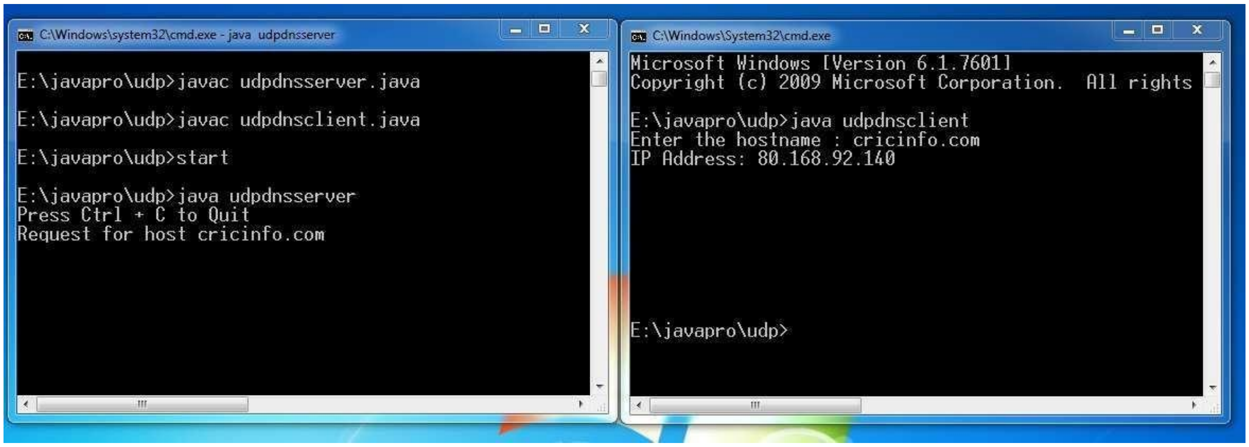
PROGRAM:

```
// UDP DNS Server -- udpdnsserver.java
import java.io.*;
import java.net.*;
public class udpdnsserver{
private static int indexOf(String[] array, String str)
{
str = str.trim();
for (int i=0; i < array.length; i++)
{
if (array[i].equals(str))
return i;}
return -1;
}
public static void main(String arg[])throws IOException
{
String[] hosts = {"yahoo.com", "gmail.com","cricinfo.com", "facebook.com"};
String[] ip = {"68.180.206.184", "209.85.148.19","80.168.92.140",
"69.63.189.16"};
System.out.println("Press Ctrl + C to Quit");
while (true)
{
DatagramSocket serversocket=new DatagramSocket(1362);
byte[] senddata = new byte[1021];
byte[] receivedata = new byte[1021];
DatagramPacket recvpack = new
DatagramPacket(receivedata, receivedata.length);
serversocket.receive(recvpack);
String sen = new String(recvpack.getData());
InetAddress ipaddress = recvpack.getAddress();
int port = recvpack.getPort();
String capsent;
System.out.println("Request for host " + sen);
if(indexOf (hosts, sen) != -1)
capsent = ip[indexOf (hosts, sen)];
else
capsent = "Host Not Found";
senddata = capsent.getBytes();
DatagramPacket pack = new DatagramPacket(senddata,
senddata.length,ipaddress,port);
serversocket.send(pack);
serversocket.close();
}
}
}
```

//UDP DNS Client -- udpdnsclient.java

```
import java.io.*;
import java.net.*;
public class udpdnsclient{
public static void main(String args[])throws IOException{
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
DatagramSocket clientsocket = new DatagramSocket();
InetAddress ipaddress;
if (args.length == 0)
ipaddress = InetAddress.getLocalHost();
else
ipaddress = InetAddress.getByName(args[0]);
byte[] senddata = new byte[1024];
byte[] receivedata = new byte[1024];
int portaddr = 1362;
System.out.print("Enter the hostname : ");
String sentence = br.readLine();
senddata = sentence.getBytes();
DatagramPacket pack = new DatagramPacket(senddata,senddata.length,
ipaddress,portaddr);
clientsocket.send(pack);
DatagramPacket recvpack =new DatagramPacket(receivedata,receivedata.length);
clientsocket.receive(recvpack);
String modified = new String(recvpack.getData());
System.out.println("IP Address: " + modified);
clientsocket.close(); }}
```

OUTPUT


C:\Windows\system32\cmd.exe - java udpdnsserver
C:\Windows\System32\cmd.exe

```
E:\javapro\udp>javac udpdnsserver.java

E:\javapro\udp>javac udpdnsclient.java

E:\javapro\udp>start

E:\javapro\udp>java udpdnsserver
Press Ctrl + C to Quit
Request for host cricinfo.com
```

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights

E:\javapro\udp>java udpdnsclient
Enter the hostname : cricinfo.com
IP Address: 80.168.92.140



E:\javapro\udp>
```

RESULT

NAME: REG NO:

Ex.No:6	**Write a code simulating ARP /RARP protocols**
DATE:	

AIM:

(A) Program for Address Resolution Protocol

ALGORITHM:

Server

1. Start the program

2. Create the socket, bind the socket created with IP address and port number and make it a listening socket.

3. Accept the connection request when it is requested by the client.

4. Server maintains the table in which IP and corresponding MAC addresses are stored.

5. Receive the IP address sent by the client.

6. Retrieve the corresponding MAC address for the IP address and send it to the client.

7. Close the connection with the client and now the server becomes a listening server waiting for the connection request from other clients

8. Stop

Client

1. Start the program

2. Create socket and establish connection with the server.

3. Get the IP address to be converted into MAC address from the user.

4. Send this IP address to server.

5. Receive the MAC address for the IP address from the server.

6. Display the received MAC address

7. Terminate the connection

PROGRAM

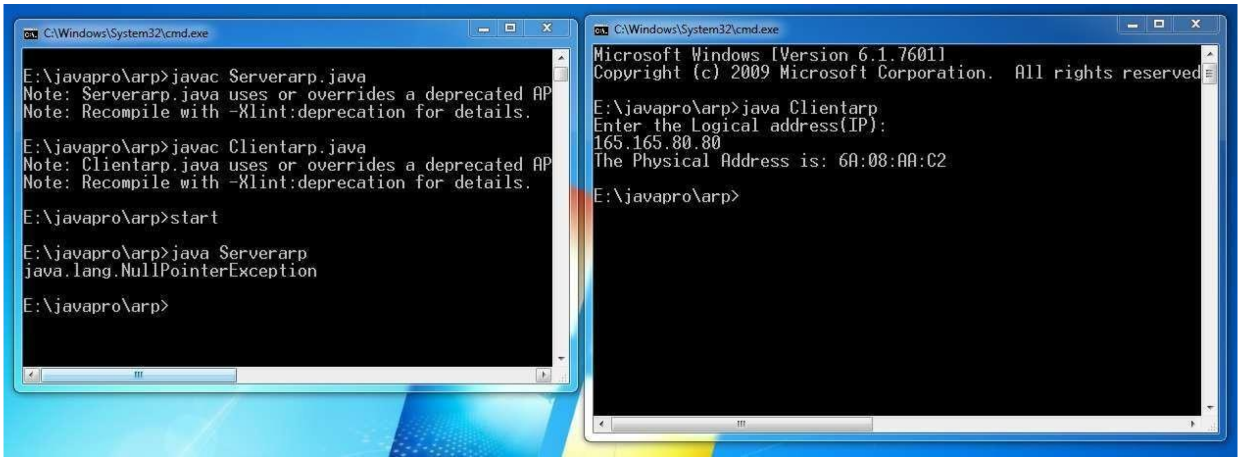
Client:

```
import java.io.*;
import java.net.*;
import java.util.*;
class Clientarp
{
public static void main(String args[])
{
try{
BufferedReader in=new BufferedReader(new InputStreamReader(System.in));
Socket clsct=new Socket("127.0.0.1",139)
DataInputStream din=new DataInputStream(clsct.getInputStream());
DataOutputStream dout=new DataOutputStream(clsct.getOutputStream());
System.out.println("Enter the Logical address(IP):");
String str1=in.readLine();
dout.writeBytes(str1+'\n';
String str=din.readLine();
System.out.println("The Physical Address is: "+str);
clsct.close();
}
catch (Exception e){
System.out.println(e);}}}
```

Server:

```
import java.io.*;
import java.net.*;
import java.util.*;
class Serverarp{
public static void main(String args[]){
try{
ServerSocket obj=new
ServerSocket(139); Socket
obj1=obj.accept();
while(true){
DataInputStream din=new DataInputStream(obj1.getInputStream());
DataOutputStream dout=new DataOutputStream(obj1.getOutputStream());
String str=din.readLine();
String ip[]={"165.165.80.80","165.165.79.1"};
String mac[]={"6A:08:AA:C2","8A:BC:E3:FA"};
for(int i=0;i<ip.length;i++)
{if(str.equals(ip[i])){
dout.writeBytes(mac[i]+'\n');
break;
}}obj.close();}}
catch(Exception e)
{System.out.println(e);}}}
```

Output:



```
E:\javapro\arp>javac Serverarp.java
Note: Serverarp.java uses or overrides a deprecated AP
Note: Recompile with -Xlint:deprecation for details.

E:\javapro\arp>javac Clientarp.java
Note: Clientarp.java uses or overrides a deprecated AP
Note: Recompile with -Xlint:deprecation for details.

E:\javapro\arp>start

E:\javapro\arp>java Serverarp
java.lang.NullPointerException

E:\javapro\arp>
```

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved

E:\javapro\arp>java Clientarp
Enter the Logical address(IP):
165.165.80.80
The Physical Address is: 6A:08:AA:C2

E:\javapro\arp>
```

(b) Program for Reverse Address Resolution Protocol (RARP) using UDP

ALGORITHM:

Server

1. Start the program.

2. Server maintains the table in which IP and corresponding MAC addresses are stored.

3. Create the datagram socket

4. Receive the datagram sent by the client and read the MAC address sent.

5. Retrieve the IP address for the received MAC address from the table.

6. Display the corresponding IP address.

7. Stop

Client

1. Start the program

2. Create datagram socket

3. Get the MAC address to be converted into IP address from the user.

4. Send this MAC address to server using UDP datagram.

5. Receive the datagram from the server and display the corresponding IP address.

6. Stop

PROGRAM:

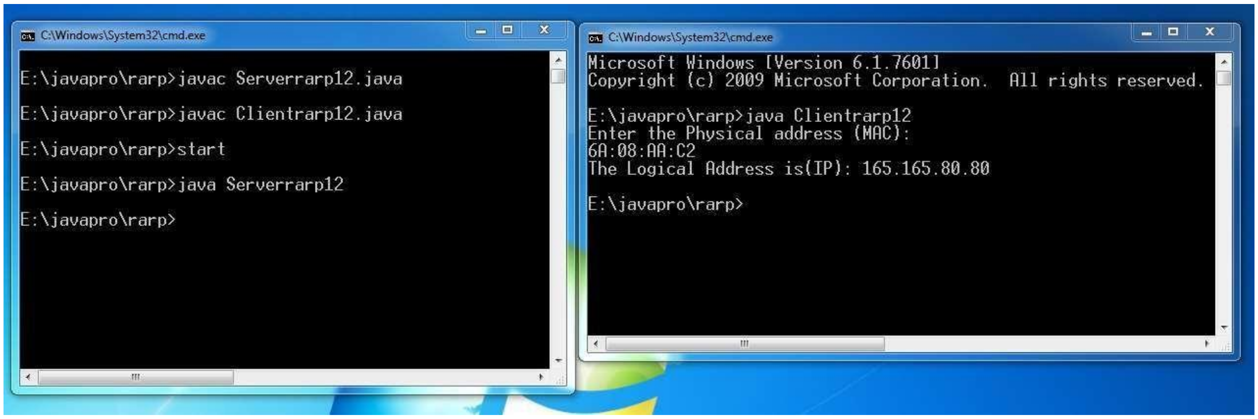
Client:

```
import java.io.*;
import java.net.*;
import java.util.*;
class Clientrarp12{
public static void main(String args[]){
try{
DatagramSocket client=new DatagramSocket();
InetAddress addr=InetAddress.getByName("127.0.0.1");
byte[] sendbyte=new byte[1024];
byte[] receivebyte=new byte[1024];
BufferedReader in=new BufferedReader(new InputStreamReader(System.in));
System.out.println("Enter the Physical address (MAC):")
String str=in.readLine(); sendbyte=str.getBytes();
DatagramPacket sender=newDatagramPacket(sendbyte,sendbyte.length,addr,1309);
client.send(sender);
DatagramPacket receiver=new DatagramPacket(receivebyte,receivebyte.length);
client.receive(receiver);
String s=new String(receiver.getData());
System.out.println("The Logical Address is(IP): "+s.trim());
client.close();}
catch(Exception e){System.out.println(e);}}}
```

Server:

```
import java.io.*;
import java.net.*;
import java.util.*;
class Serverrarp12{
public static void main(String args[]){
try{
DatagramSocket server=new DatagramSocket(1309);
while(true){
byte[] sendbyte=new byte[1024];
byte[] receivebyte=new byte[1024];
DatagramPacket receiver=new DatagramPacket(receivebyte,receivebyte.length);
server.receive(receiver);
String str=new String(receiver.getData());
String s=str.trim();
InetAddress addr=receiver.getAddress();
int port=receiver.getPort();
String ip[]={"165.165.80.80","165.165.79.1"};
String mac[]={"6A:08:AA:C2","8A:BC:E3:FA"};
for(int i=0;i<ip.length;i++){
if(s.equals(mac[i]))
{sendbyte=ip[i].getBytes();
DatagramPacket sender = new
DatagramPacket(sendbyte,sendbyte.length,addr,port);
server.send(sender);

break;}}break;}}}catch(Exception  e){System.out.println(e);}}}
```

Output:



```
E:\javapro\rarp>javac Serverrarp12.java

E:\javapro\rarp>javac Clientrarp12.java

E:\javapro\rarp>start

E:\javapro\rarp>java Serverrarp12

E:\javapro\rarp>
```

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

E:\javapro\rarp>java Clientrarp12
Enter the Physical address (MAC):
6A:08:AA:C2
The Logical Address is(IP): 165.165.80.80

E:\javapro\rarp>
```

RESULT:

Ex.No: 7	**Traffic Analysis using Wireshark.**
DATE:	

AIM:

THEORY:

Network traffic analysis is the routine task of various job roles, such as network administrator, network defenders, incident responders and others. Wireshark plays a vital role during the traffic analysis;

This article covers the traffic analysis of the most common network protocols, for example, ICMP, ARP, HTTPS, TCP, etc. **Capture filters with protocol header values** Wireshark comes with several capture and display filters. But a user can create display filters using protocol header values as well. Use this technique to analyze traffic efficiently.

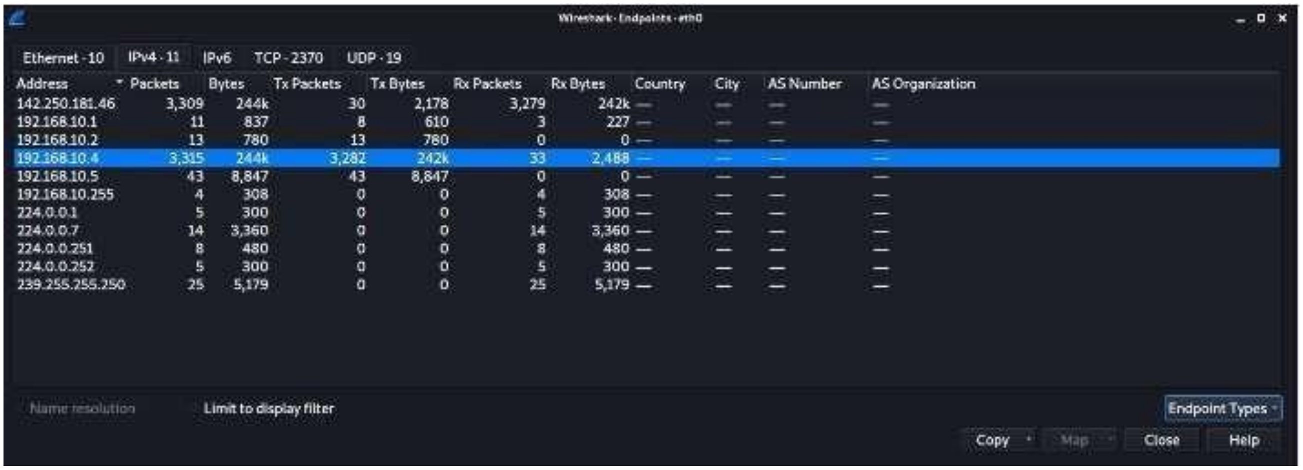
proto[offset:size(optional)]=value

Following the above syntax, it is easy to create a dynamic capture filter, where:

- proto = desired protocol
- offset = header value
- size = data length
- value = data you want to find

Analyzing endpoints

This feature comes in handy to determine the endpoint generating the highest volume or abnormal traffic in the network. To analyze the endpoints between two communication devices, do the following:Capture traffic and select the packet whose endpoint you wish to check. -> Click Statistics menu -> Select Endpoints.The most traffic-intensive endpoint, as seen in the picture below, is 192.168.10.4.

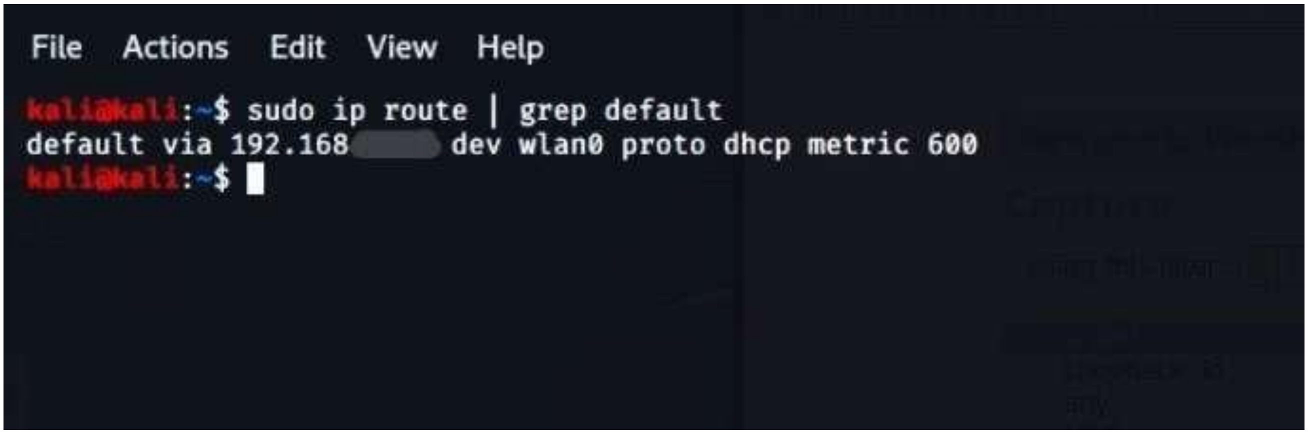


Wireshark · Endpoints · eth0

Address	Packets	Bytes	Tx Packets	Tx Bytes	Rx Packets	Rx Bytes	Country	City	AS Number	AS Organization
142.250.181.46	3,309	244k	30	2,178	3,279	242k	—	—	—	—
192.168.10.1	11	837	8	610	3	227	—	—	—	—
192.168.10.2	13	780	13	780	0	0	—	—	—	—
192.168.10.4	3,315	244k	3,282	242k	33	2,488	—	—	—	—
192.168.10.5	43	8,847	43	8,847	0	0	—	—	—	—
192.168.10.255	4	308	0	0	4	308	—	—	—	—
224.0.0.1	5	300	0	0	5	300	—	—	—	—
224.0.0.7	14	3,360	0	0	14	3,360	—	—	—	—
224.0.0.251	8	480	0	0	8	480	—	—	—	—
224.0.0.252	5	300	0	0	5	300	—	—	—	—
239.255.255.250	25	5,179	0	0	25	5,179	—	—	—	—

ARP traffic analysis

Address resolution protocol (ARP) generally uses to find the MAC address of the target machine. In this demo, let's try capturing and analyzing ARP traffic.First things first, know the target machine IP. In our case, it's going to be the default gateway address.

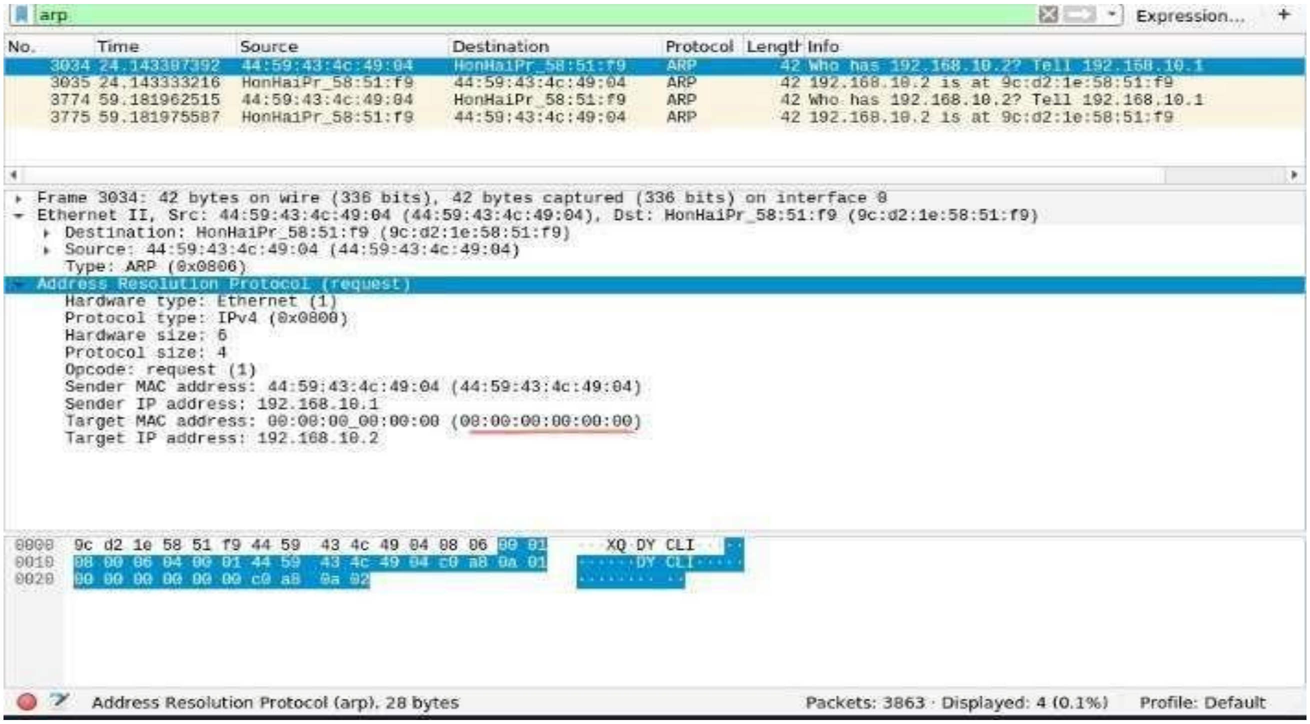


```
kali@kali:~$ sudo ip route | grep default
default via 192.168[redacted] dev wlan0 proto dhcp metric 600
kali@kali:~$
```

Start Wireshark data capturing, and ping the default gateway address -> Now, let's analyze what happens after removing the ARP entry and pinging a new IP address in the meantime.

Analyze an ARP Request

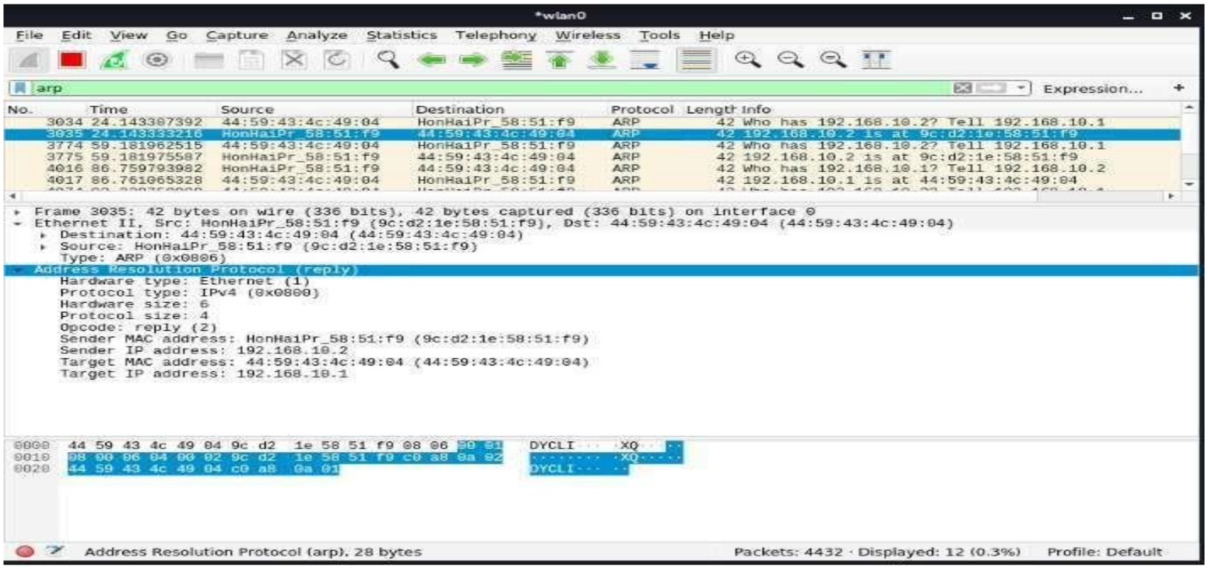
Using the *'arp'* filter, analyze the captured traffic in Wireshark.Observe the packet request details from Ethernet and ARP; observe the source and destination IP and sender MAC and IP address.Monitor the victim's MAC address. Since the destination MAC address is unavailable at the request packet stage, the victim's MAC address is zero, and the destination IP is the local system IP address.



```
arp
No.   Time               Source              Destination         Protocol  Length Info
 3034 24.143387392  44:59:43:4c:49:04   HonHaiPr_58:51:f9   ARP        42 Who has 192.168.10.2? Tell 192.168.10.1
 3035 24.143333216  HonHaiPr_58:51:f9   44:59:43:4c:49:04   ARP        42 192.168.10.2 is at 9c:d2:1e:58:51:f9
 3774 59.181962515  44:59:43:4c:49:04   HonHaiPr_58:51:f9   ARP        42 Who has 192.168.10.2? Tell 192.168.10.1
 3775 59.181975587  HonHaiPr_58:51:f9   44:59:43:4c:49:04   ARP        42 192.168.10.2 is at 9c:d2:1e:58:51:f9

▸ Frame 3034: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
▾ Ethernet II, Src: 44:59:43:4c:49:04 (44:59:43:4c:49:04), Dst: HonHaiPr_58:51:f9 (9c:d2:1e:58:51:f9)
  ▸ Destination: HonHaiPr_58:51:f9 (9c:d2:1e:58:51:f9)
  ▸ Source: 44:59:43:4c:49:04 (44:59:43:4c:49:04)
    Type: ARP (0x0806)
▾ Address Resolution Protocol (request)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (1)
    Sender MAC address: 44:59:43:4c:49:04 (44:59:43:4c:49:04)
    Sender IP address: 192.168.10.1
    Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
    Target IP address: 192.168.10.2

0000  9c d2 1e 58 51 f9 44 59  43 4c 49 04 08 06 00 01   ···XQ·DY CLI····
0010  08 00 06 04 00 01 44 59  43 4c 49 04 c0 a8 0a 01   ······DY CLI····
0020  00 00 00 00 00 00 c0 a8  0a 02                     ··········

Address Resolution Protocol (arp), 28 bytes      Packets: 3863 · Displayed: 4 (0.1%)      Profile: Default
```

Analyze an ARP Response

Observe the packet replay details from Ethernet and ARP; observe the change in source and destination IP and MAC addresses.The destination and source MAC address are switched in the response packet. Everything is similar as before, except the target MAC address, which was all zeroes before. Now, that has turned into your MAC address.



*wlan0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Expression...

```
arp
No.   Time               Source              Destination         Protocol  Length Info
 3034 24.143387392  44:59:43:4c:49:04   HonHaiPr_58:51:f9   ARP        42 Who has 192.168.10.2? Tell 192.168.10.1
 3035 24.143333216  HonHaiPr_58:51:f9   44:59:43:4c:49:04   ARP        42 192.168.10.2 is at 9c:d2:1e:58:51:f9
 3774 59.181962515  44:59:43:4c:49:04   HonHaiPr_58:51:f9   ARP        42 Who has 192.168.10.2? Tell 192.168.10.1
 3775 59.181975587  HonHaiPr_58:51:f9   44:59:43:4c:49:04   ARP        42 192.168.10.2 is at 9c:d2:1e:58:51:f9
 4016 86.759793982  HonHaiPr_58:51:f9   44:59:43:4c:49:04   ARP        42 Who has 192.168.10.1? Tell 192.168.10.2
 4017 86.761065328  44:59:43:4c:49:04   HonHaiPr_58:51:f9   ARP        42 192.168.10.1 is at 44:59:43:4c:49:04

▸ Frame 3035: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
▾ Ethernet II, Src: HonHaiPr_58:51:f9 (9c:d2:1e:58:51:f9), Dst: 44:59:43:4c:49:04 (44:59:43:4c:49:04)
  ▸ Destination: 44:59:43:4c:49:04 (44:59:43:4c:49:04)
  ▸ Source: HonHaiPr_58:51:f9 (9c:d2:1e:58:51:f9)
    Type: ARP (0x0806)
▾ Address Resolution Protocol (reply)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: reply (2)
    Sender MAC address: HonHaiPr_58:51:f9 (9c:d2:1e:58:51:f9)
    Sender IP address: 192.168.10.2
    Target MAC address: 44:59:43:4c:49:04 (44:59:43:4c:49:04)
    Target IP address: 192.168.10.1

0000  44 59 43 4c 49 04 9c d2  1e 58 51 f9 08 06 00 01   DYCLI···  ·XQ····
0010  08 00 06 04 00 02 9c d2  1e 58 51 f9 c0 a8 0a 02   ········  ·XQ····
0020  44 59 43 4c 49 04 c0 a8  0a 01                     DYCLI···  ··

Address Resolution Protocol (arp), 28 bytes      Packets: 4432 · Displayed: 12 (0.3%)      Profile: Default
```

Result:

NAME: REG NO:

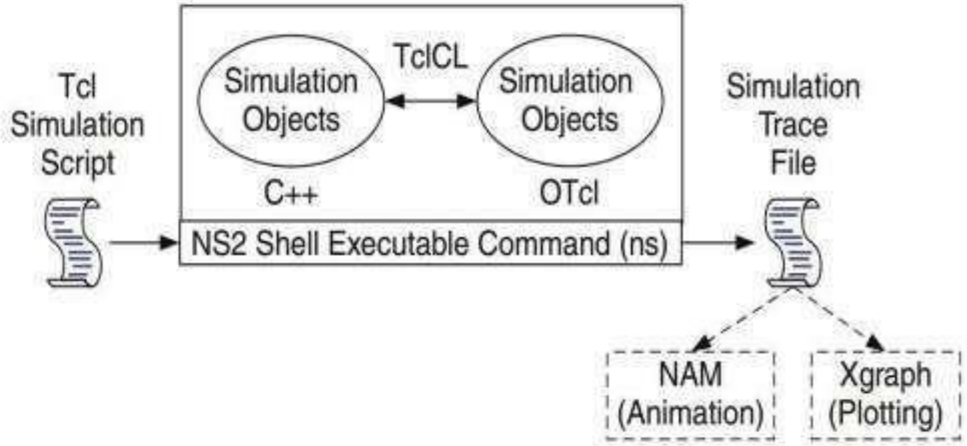
Ex.No: 8	**Study of Network simulator (NS) and Simulation of Congestion Control Algorithms using NS**
DATE:	

AIM:

THEORY:

Network Simulator (Version 2), widely known as NS2, is simply an event driven simulation tool that has proved useful in studying the dynamic nature of communication networks. Simulation of wired as well as wireless network functions and protocols (e.g., routing algorithms, TCP, UDP) can be done using NS2. In general, NS2 provides users with a way of specifying such network protocols and simulating their corresponding behaviors. Due to its flexibility and modular nature, NS2 has gained constant popularity in thenetworking research community since its birth in 1989.

Basic Architecture of NS2:



The above figure shows the basic architecture of NS2. NS2 provides users with an executable command ns which takes on input argument, the name of a Tcl simulation scripting file. Users are feeding the name of a Tcl simulation script (which sets up a simulation) as an input argument of an NS2 executable command ns.

In most cases, a simulation trace file is created, and is used to plot graph and/or to create animation. NS2 consists of two key languages: C++ and Object-oriented Tool Command Language (OTcl). While the C++ defines the internal mechanism (i.e., a backend) of the simulation objects, the

OTcl sets up simulation by assembling and configuring the objects as well as scheduling discrete events (i.e., a frontend).

The C++ and the OTcl are linked together using TclCL. Mapped to a C++ object, variables in the OTcl domains are sometimes referred to as handles. Conceptually, a handle (e.g., n as a Node handle) is just a string (e.g.,_o10) in the OTcl domain, and does not contain any functionality. instead, the functionality (e.g., receiving a packet) is defined in the mapped C++ object (e.g., of class Connector). In the OTcl domain, a handle acts as a frontend which interacts with users and other OTcl objects. It may defines its own procedures and variables to facilitate the interaction. Note that the member procedures and variables in the OTcl domain are called instance procedures (instprocs) and instance variables (instvars), respectively

NS2 provides a large number of built-in C++ objects. It is advisable to use these C++ objects to set up a simulation using a Tcl simulation script. However, advance users may find these objects insufficient. They need to develop their own C++ objects, and use a OTcl configuration interface to put together these objects. After simulation, NS2 outputs either text-based or animation-based simulation results. To interpret these results graphically and interactively, tools such as NAM (Network AniMator) and XGraph are used. To analyze a particular behaviour of the network, users can extract a relevant subset of text-based data and transform it to a more conceivable presentation

Tcl scripting:

- Tcl is a general purpose scripting language. [Interpreter]
- Tcl runs on most of the platforms such as Unix, Windows, and Mac.
- The strength of Tcl is its simplicity.
- It is not necessary to declare a data type for variable prior to the usage

Basics of TCL

Syntax: command arg1 arg2 arg3

Hello World!

```
puts stdout{Hello, World!} Hello, World!
```

Variables

Command Substitution

```
set a 5 set len [string length foobar]
set b $a set len [expr [string length foobar] + 9]
```

Wired TCL Script Components

Create the event scheduler
Open new files & turn on the tracing
Create the nodes Setup the links Configure the traffic type (e.g., TCP, UDP, etc)
Set the time of traffic generation (e.g., CBR, FTP)
Terminate the simulation

NS2 Simulator Preliminaries.

1. Initialization and termination aspects of the ns simulator.
2. Definition of network nodes, links, queues and topology.
3. Definition of agents and of applications.
4. The nam visualization tool.
5. Tracing and random variables

Initialization and Termination of TCL Script in NS-2

An ns simulation starts with the command

```
set ns [new Simulator]
```

Which is thus the first line in the tcl script. This line declares a new variable as using the set command, you can call this variable as you wish, In general people declares it as ns because it is an instance of the Simulator class, so an object the code[new Simulator] is indeed the installation of the class Simulator using the reserved word new.

In order to have output files with data on the simulation (trace files) or files used for visualization (nam files), we need to create the files using ―open command:

#Open the Trace file

```
set tracefile1 [open out.tr w]
$ns trace-all $tracefile1
```

#Open the NAM trace file

```
set namfile [open out.nam w]
$ns namtrace-all $namfile
```

The above creates a dta trace file called out.tr and a nam visualization trace file called out.nam. Within the tcl script, these files are not called explicitly by their names, but instead by pointers that are declared above and called ―tracefile1 and ―namfile respectively. Remark that they begins with a # symbol. The second line open the file ―out.tr to be used for writing, declared with the letter ―w. The third line uses a simulator method called trace-all that have as parameter the name of the file where the traces will go.

Define a "finish" procedure

```
Proc finish { } {
global ns tracefile1 namfile
 $ns flush-trace
Close $tracefile1
Close $namfile
```

NAME: REG NO:

```
Exec nam out.nam &
Exit 0
}
```

Definition of a network of links and nodes

The way to define a node is

set n0 [$ns node]

Once we define several nodes, we can define the links that connect them. An example of a definition of a link is:

$ns duplex-link $n0 $n2 10Mb 10ms DropTail

Which means that $n0 and $n2 are connected using a bi-directional link that has 10ms of propagation delay and a capacity of 10Mb per sec for each direction.

To define a directional link instead of a bi-directional one, we should replace ―duplex-link by ―simplex-link. In ns, an output queue of a node is implemented as a part of each link whose input is that node. We should also define the buffer capacity of the queue related to each link. An example would be:

#set Queue Size of link (n0-n2) to 20
$ns queue-limit $n0 $n2 20

FTP over TCP

TCP is a dynamic reliable congestion control protocol. It uses Acknowledgements created by the destination to know whether packets are well received.
There are number variants of the TCP protocol, such as Tahoe, Reno, NewReno, Vegas. The type of agent appears in the first line:

set tcp [new Agent/TCP]

The command **$ns attach-agent $n0 $tcp** defines the source node of the tcp connection.

The command **set sink [new Agent /TCPSink]**Defines the behavior of the destination node of TCP and assigns to it a pointer called sink.

```
#Setup a UDP connection

set udp [new Agent/UDP]
$ns attach-agent $n1 $udp
set null [new Agent/Null]
$ns attach-agent $n5 $null
$ns connect $udp $null
$udp set fid_2
```

#setup a CBR over UDP connection

The below shows the definition of a CBR application using a UDP agent

The command **$ns attach-agent $n4 $sink** defines the destination node. The command **$ns connect $tcp $sink** finally makes the TCP connection between the source and destination nodes.

```
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set packetsize_ 100
$cbr set rate_ 0.01Mb
$cbr set random_ false
```

TCP has many parameters with initial fixed defaults values that can be changed if mentioned explicitly. For example, the default TCP packet size has a size of 1000bytes.This can be changed to another value, say 552bytes, using the command **$tcp set packetSize_ 552**.

When we have several flows, we may wish to distinguish them so that we can identify them with different colors in the visualization part. This is done by the command **$tcp set fid_ 1** that assigns to the TCP connection a flow identification of ―1.We shall later give the flow identification of ―2‖ to the UDP connection.

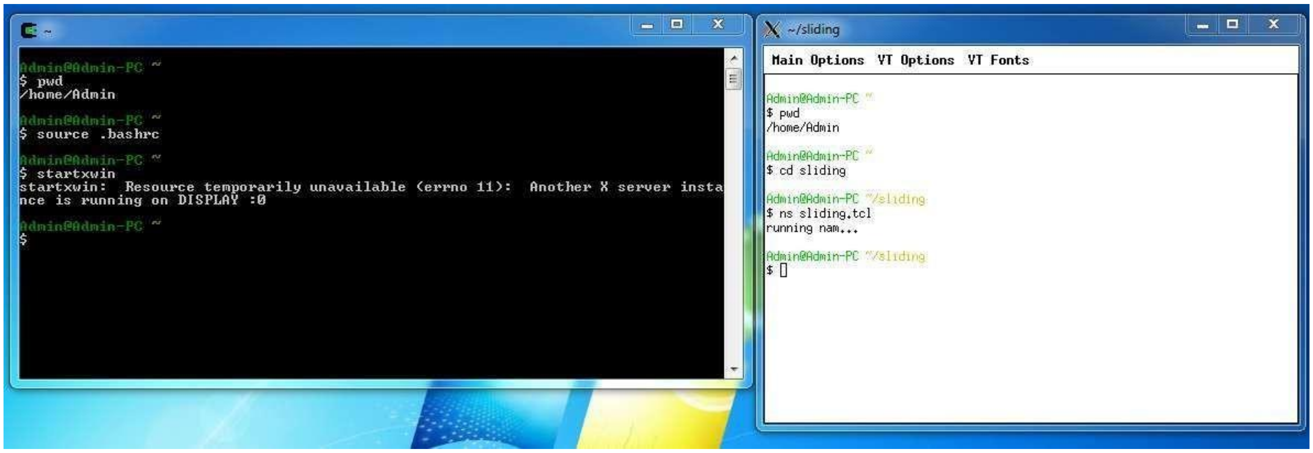
Simulation of congestion control using sliding window protocol

Sliding.tcl

```
# sliding window mechanism with some features
# such as labeling, annotation, nam-graph, and window size monitoring
set ns [new Simulator]
set n0 [$ns node]
set n1 [$ns node]
$ns at 0.0 "$n0 label Sender"
$ns at 0.0 "$n1 label Receiver"
set nf [open sliding.nam w]
$ns namtrace-all $nf
set f [open sliding.tr w]
$ns trace-all $f
$ns duplex-link $n0 $n1 0.2Mb 200ms DropTail
$ns duplex-link-op $n0 $n1 orient right
$ns queue-limit $n0 $n1 10
Agent/TCP set nam_tracevar_ true
set tcp [new Agent/TCP]
$tcp set windowInit_ 4
$tcp set maxcwnd_ 4
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n1 $sink
$ns connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns add-agent-trace $tcp tcp
$ns monitor-agent-trace $tcp
$tcp tracevar cwnd_
$ns at 0.1 "$ftp start"
$ns at 3.0 "$ns detach-agent $n0 $tcp ; $ns detach-agent $n1 $sink"
$ns at 3.5 "finish"
$ns at 0.0 "$ns trace-annotate \"Sliding Window with window size 4 (normal operation)\""
$ns at 0.05 "$ns trace-annotate \"FTP starts at 0.1\""
$ns at 0.11 "$ns trace-annotate \"Send Packet_0,1,2,3\""
$ns at 0.34 "$ns trace-annotate \"Receive Ack_0,1,2,3\""
$ns at 0.56 "$ns trace-annotate \"Send Packet_4,5,6,7\""
$ns at 0.79 "$ns trace-annotate \"Receive Ack_4,5,6,7\""
$ns at 0.99 "$ns trace-annotate \"Send Packet_8,9,10,11\""
$ns at 1.23 "$ns trace-annotate \"Receive Ack_8,9,10,11 \""
$ns at 1.43 "$ns trace-annotate \"Send Packet_12,13,14,15\""
```

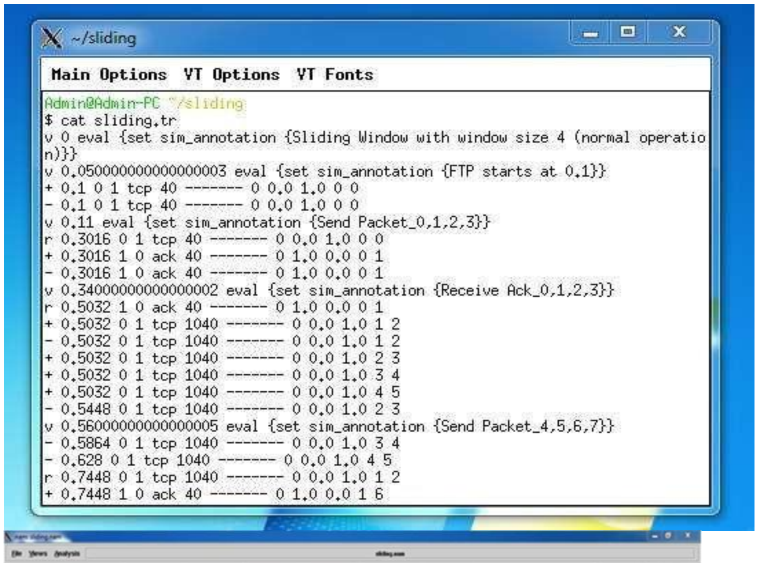
```
$ns at 1.67 "$ns trace-annotate \"Receive Ack_12,13,14,15\""
$ns at 1.88 "$ns trace-annotate \"Send Packet_16,17,18,19\""
$ns at 2.11 "$ns trace-annotate \"Receive Ack_16,17,18,19\""
$ns at 2.32 "$ns trace-annotate \"Send Packet_20,21,22,23\""
$ns at 2.56 "$ns trace-annotate \"Receive Ack_24,25,26,27\""
$ns at 2.76 "$ns trace-annotate \"Send Packet_28,29,30,31\""
$ns at 3.00 "$ns trace-annotate \"Receive Ack_28\""
$ns at 3.1 "$ns trace-annotate \"FTP stops\""

proc finish {} {
global ns
$ns flush-trace
# close $nf
puts "running nam..."
exec nam sliding.nam &
exit 0
}
$ns run
```

OUTPUT:

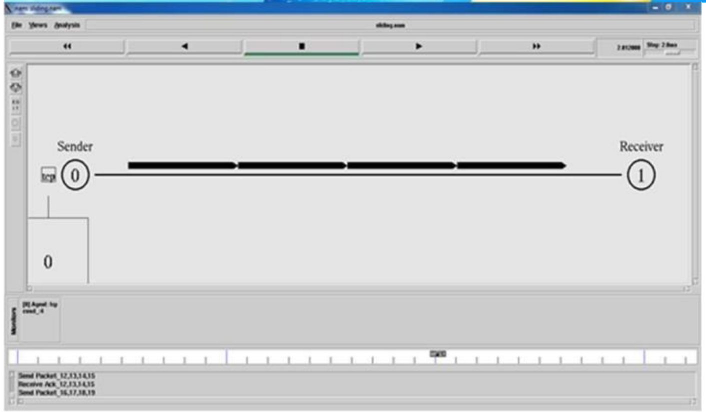


```
Admin@Admin-PC ~
$ pwd
/home/Admin

Admin@Admin-PC ~
$ source .bashrc

Admin@Admin-PC ~
$ startxwin
startxwin:  Resource temporarily unavailable (errno 11):  Another X server insta
nce is running on DISPLAY :0

Admin@Admin-PC ~
$
```

```
Admin@Admin-PC ~
$ pwd
/home/Admin

Admin@Admin-PC ~
$ cd sliding

Admin@Admin-PC ~/sliding
$ ns sliding.tcl
running nam...

Admin@Admin-PC ~/sliding
$ 
```



```
Admin@Admin-PC ~/sliding
$ cat sliding.tr
v 0 eval {set sim_annotation {Sliding Window with window size 4 (normal operatio
n)}}
v 0.050000000000000003 eval {set sim_annotation {FTP starts at 0.1}}
+ 0.1 0 1 tcp 40 ------- 0 0.0 1.0 0 0
- 0.1 0 1 tcp 40 ------- 0 0.0 1.0 0 0
v 0.11 eval {set sim_annotation {Send Packet_0,1,2,3}}
r 0.3016 0 1 tcp 40 ------- 0 0.0 1.0 0 0
+ 0.3016 1 0 ack 40 ------- 0 1.0 0.0 0 1
- 0.3016 1 0 ack 40 ------- 0 1.0 0.0 0 1
v 0.34000000000000002 eval {set sim_annotation {Receive Ack_0,1,2,3}}
r 0.5032 1 0 ack 40 ------- 0 1.0 0.0 0 1
+ 0.5032 0 1 tcp 1040 ------- 0 0.0 1.0 1 2
- 0.5032 0 1 tcp 1040 ------- 0 0.0 1.0 1 2
+ 0.5032 0 1 tcp 1040 ------- 0 0.0 1.0 2 3
+ 0.5032 0 1 tcp 1040 ------- 0 0.0 1.0 3 4
+ 0.5032 0 1 tcp 1040 ------- 0 0.0 1.0 4 5
- 0.5448 0 1 tcp 1040 ------- 0 0.0 1.0 2 3
v 0.56000000000000005 eval {set sim_annotation {Send Packet_4,5,6,7}}
- 0.5864 0 1 tcp 1040 ------- 0 0.0 1.0 3 4
- 0.628 0 1 tcp 1040 ------- 0 0.0 1.0 4 5
r 0.7448 0 1 tcp 1040 ------- 0 0.0 1.0 1 2
+ 0.7448 1 0 ack 40 ------- 0 1.0 0.0 1 6
```



```
Send Packet_12,13,14,15
Receive Ack_12,13,14,15
Send Packet_16,17,18,19
```

Result:

NAME: REG NO:

Ex.No: 9	**Study of TCP/UDP performance using Simulation tool.**
DATE:	

AIM:

Algorithm

1. Create a Simulator object.
2. Set routing as dynamic.
3. Open the trace and nam trace files.
4. Define the finish procedure.
5. Create nodes and the links between them.
6. Create the agents and attach them to the nodes.
7. Create the applications and attach them to the tcp agent.
8. Connect tcp and tcp sink.
9. Run the simulation

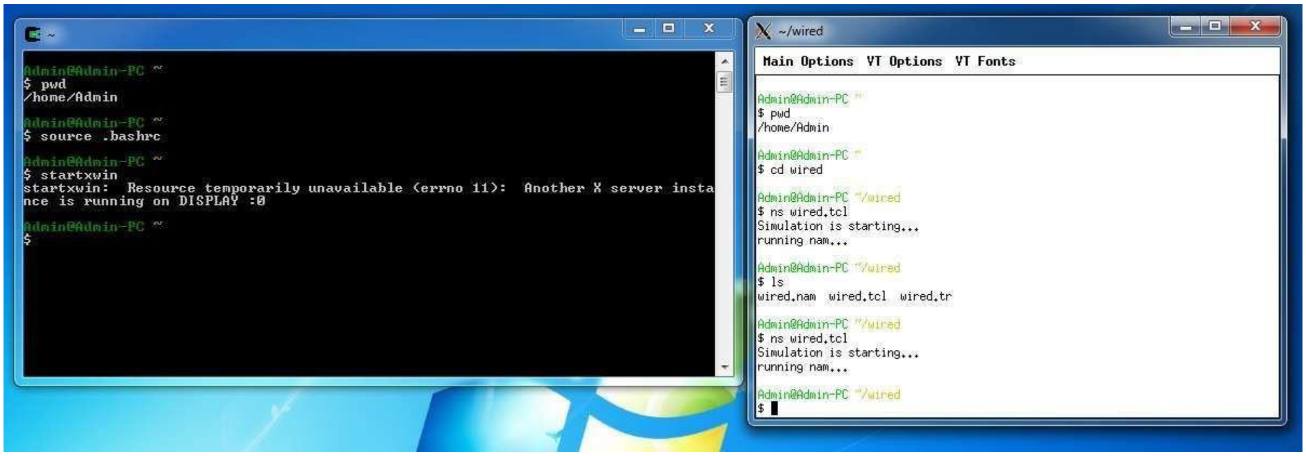
NAME: REG NO:

PROGRAM:
wired.tcl

```
#create a simulator object
set ns [new Simulator]

#create a trace file, this file is for logging purpose
set tracefile [open wired.tr w]
$ns trace-all $tracefile

#create a animation infomration or NAM file creation
set namfile [open wired.nam w]
$ns namtrace-all $namfile

#create nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]

#creation of link between nodes with DropTail Queue
#Droptail means Dropping the tail.
$ns duplex-link $n0 $n1 5Mb 2ms DropTail
$ns duplex-link $n2 $n1 10Mb 5ms DropTail
$ns duplex-link $n1 $n4 3Mb 10ms DropTail
$ns duplex-link $n4 $n3 100Mb 2ms DropTail
$ns duplex-link $n4 $n5 4Mb 10ms DropTail

#creation of Agents
#node 0 to Node 3
set udp [new Agent/UDP]
set null [new Agent/Null]
$ns attach-agent $n0 $udp
$ns attach-agent $n3 $null
$ns connect $udp $null

#creation of TCP Agent
set tcp [new Agent/TCP]
set sink [new Agent/TCPSink]
$ns attach-agent $n2 $tcp
$ns attach-agent $n5 $sink
$ns connect $tcp $sink
```

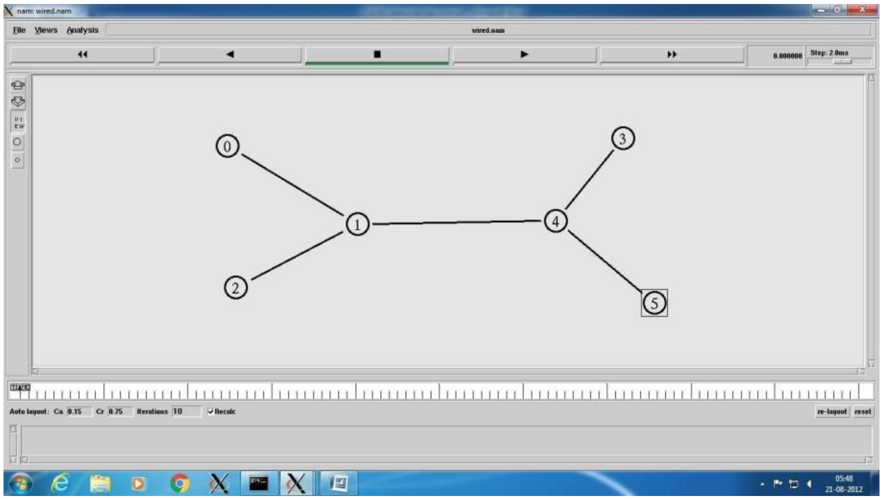
```
#creation of Application CBR, FTP
#CBR - Constant Bit Rate (Example nmp3 files that have a CBR or 192kbps, 320kbps, etc.)
#FTP - File Transfer Protocol (Ex: To download a file from a network)
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp

set ftp [new Application/FTP]
$ftp attach-agent $tcp

#Start the traffic
$ns at 1.0 "$cbr start"
$ns at 2.0 "$ftp start"

$ns at 10.0 "finish"

#the following procedure will be called at 10.0 seconds
proc finish {} {
 global ns tracefile namfile
 $ns  flush-trace
 close $tracefile
 close $namfile
 puts "running nam..."
exec nam wired.nam &
exit 0
}

puts "Simulation is starting..."
$ns run
```

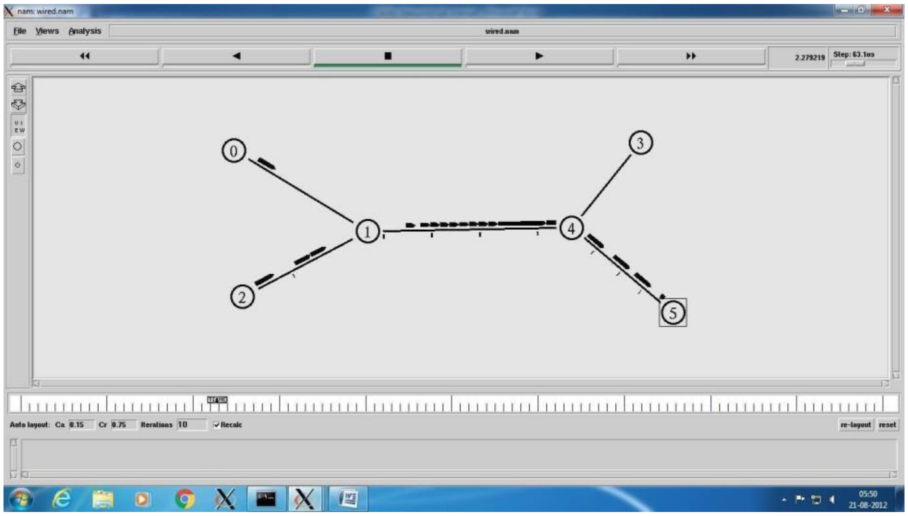
OUTPUT:



```
Admin@Admin-PC ~
$ pwd
/home/Admin

Admin@Admin-PC ~
$ source .bashrc

Admin@Admin-PC ~
$ startxwin
startxwin:  Resource temporarily unavailable (errno 11):  Another X server insta
nce is running on DISPLAY :0

Admin@Admin-PC ~
$
```

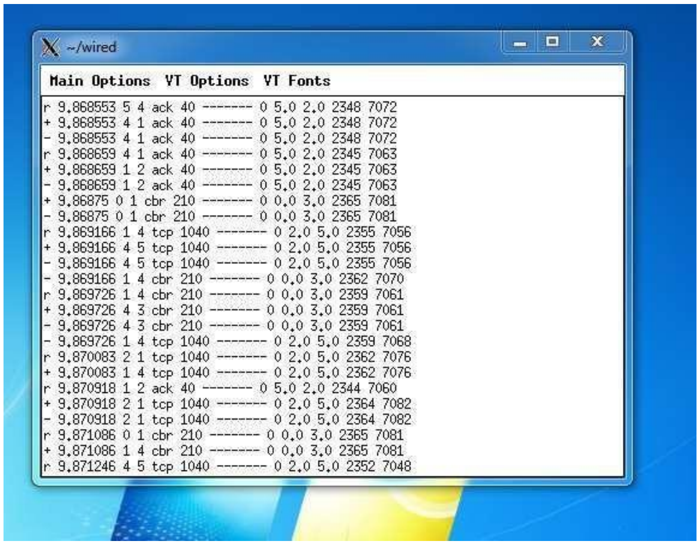
```
Admin@Admin-PC ~
$ pwd
/home/Admin

Admin@Admin-PC ~
$ cd wired

Admin@Admin-PC ~/wired
$ ns wired.tcl
Simulation is starting...
running nam...

Admin@Admin-PC ~/wired
$ ls
wired.nam  wired.tcl  wired.tr

Admin@Admin-PC ~/wired
$ ns wired.tcl
Simulation is starting...
running nam...

Admin@Admin-PC ~/wired
$
```





NAME: REG NO:

wired.tr



```
r 9.868553 5 4 ack 40 ------- 0 5.0 2.0 2348 7072
+ 9.868553 4 1 ack 40 ------- 0 5.0 2.0 2348 7072
- 9.868553 4 1 ack 40 ------- 0 5.0 2.0 2348 7072
r 9.868659 4 1 ack 40 ------- 0 5.0 2.0 2345 7063
+ 9.868659 1 2 ack 40 ------- 0 5.0 2.0 2345 7063
- 9.868659 1 2 ack 40 ------- 0 5.0 2.0 2345 7063
+ 9.86875 0 1 cbr 210 ------- 0 0.0 3.0 2365 7081
- 9.86875 0 1 cbr 210 ------- 0 0.0 3.0 2365 7081
r 9.869166 1 4 tcp 1040 ------- 0 2.0 5.0 2355 7056
+ 9.869166 4 5 tcp 1040 ------- 0 2.0 5.0 2355 7056
- 9.869166 4 5 tcp 1040 ------- 0 2.0 5.0 2355 7056
- 9.869166 1 4 cbr 210 ------- 0 0.0 3.0 2362 7070
r 9.869726 1 4 cbr 210 ------- 0 0.0 3.0 2359 7061
+ 9.869726 4 3 cbr 210 ------- 0 0.0 3.0 2359 7061
- 9.869726 4 3 cbr 210 ------- 0 0.0 3.0 2359 7061
- 9.869726 1 4 tcp 1040 ------- 0 2.0 5.0 2359 7068
r 9.870083 2 1 tcp 1040 ------- 0 2.0 5.0 2362 7076
+ 9.870083 1 4 tcp 1040 ------- 0 2.0 5.0 2362 7076
r 9.870918 1 2 ack 40 ------- 0 5.0 2.0 2344 7060
+ 9.870918 2 1 tcp 1040 ------- 0 2.0 5.0 2364 7082
- 9.870918 2 1 tcp 1040 ------- 0 2.0 5.0 2364 7082
r 9.871086 0 1 cbr 210 ------- 0 0.0 3.0 2365 7081
+ 9.871086 1 4 cbr 210 ------- 0 0.0 3.0 2365 7081
r 9.871246 4 5 tcp 1040 ------- 0 2.0 5.0 2352 7048
```

RESULT:

Ex.NO:10(a)	**Simulation of Distance Vector/ Link State Routing algorithm.**
DATE:	

AIM:

SOFTWARE REQUIRED:
NS-2

THEORY:
Distance Vector Routing is one of the routing algorithm in a Wide Area Network for computing shortest path between source and destination. The Router is one main devices used in a wide area network. The main task of the router is Routing. It forms the routing table and delivers the packets depending upon the routes in the table-either directly or via an intermediate devices.
Each router initially has information about its all neighbors. Then this information will be shared among nodes.

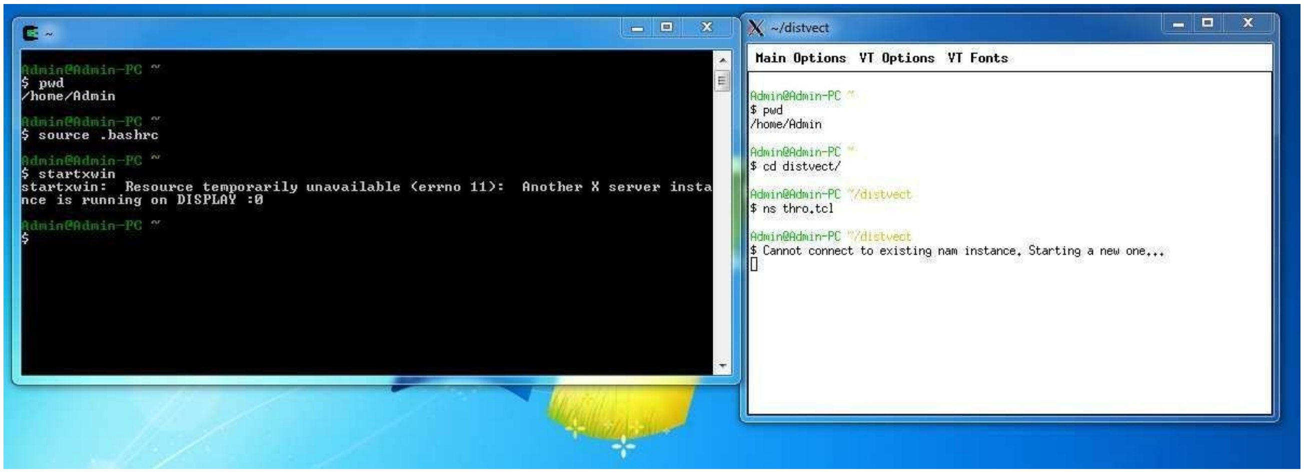
ALGORITHM:
1. Create a simulator object
2. Define different colors for different data flows
3. Open a nam trace file and define finish procedure then close the trace file, and execute nam on trace file.
4. Create n number of nodes using for loop
5. Create duplex links between the nodes
6. Setup UDP Connection between n(0) and n(5)
7. Setup another UDP connection between n(1) and n(5)
8. Apply CBR Traffic over both UDP connections
9. Choose distance vector routing protocol to transmit data from sender to receiver.
10. Schedule events and run the program.

PROGRAM:

```
set ns [new Simulator]
set nr [open thro.tr w]
$ns trace-all $nr
set nf [open thro.nam w]
$ns namtrace-all $nf
proc finish { } {
global ns nr nf
$ns flush-trace
close $nf
close $nr
exec nam thro.nam &
exit 0
}
for { set i 0 } { $i < 12} { incr i 1 } {
set n($i) [$ns node]}
for {set i 0} {$i < 8} {incr i} {
$ns duplex-link $n($i) $n([expr $i+1]) 1Mb 10ms DropTail }
$ns duplex-link $n(0) $n(8) 1Mb 10ms DropTail
$ns duplex-link $n(1) $n(10) 1Mb 10ms DropTail

$ns duplex-link $n(0) $n(9) 1Mb 10ms DropTail
$ns duplex-link $n(9) $n(11) 1Mb 10ms DropTail
$ns duplex-link $n(10) $n(11) 1Mb 10ms DropTail
$ns duplex-link $n(11) $n(5) 1Mb 10ms DropTail
set udp0 [new Agent/UDP]
$ns attach-agent $n(0) $udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0
set null0 [new Agent/Null]
$ns attach-agent $n(5) $null0
$ns connect $udp0 $null0
set udp1 [new Agent/UDP]
$ns attach-agent $n(1) $udp1
set cbr1 [new Application/Traffic/CBR]
$cbr1 set packetSize_ 500
$cbr1 set interval_ 0.005
$cbr1 attach-agent $udp1
set null0 [new Agent/Null]
$ns attach-agent $n(5) $null0
$ns connect $udp1 $null0
```

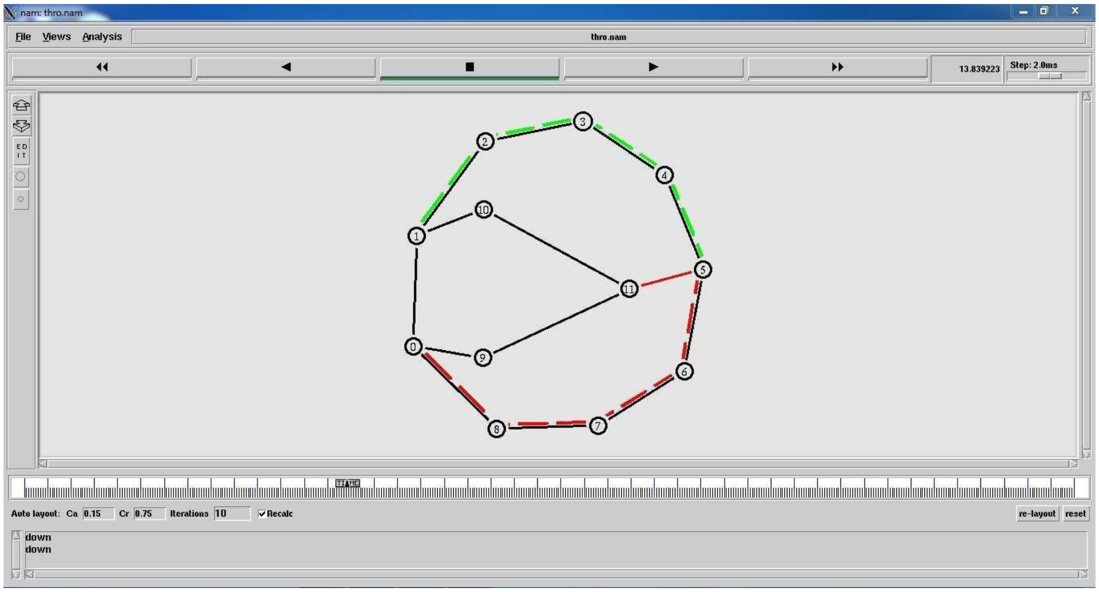
```
$ns rtproto DV
$ns rtmodel-at 10.0 down $n(11) $n(5)
$ns rtmodel-at 15.0 down $n(7) $n(6)
$ns rtmodel-at 30.0 up $n(11) $n(5)
$ns rtmodel-at 20.0 up $n(7) $n(6)
$udp0 set fid_ 1
$udp1 set fid_ 2
$ns color 1 Red
$ns color 2 Green
$ns at 1.0 "$cbr0 start"
$ns at 2.0 "$cbr1 start"
$ns at 45 "finish"
$ns run
```

OUTPUT:



```
Admin@Admin-PC ~
$ pwd
/home/Admin

Admin@Admin-PC ~
$ source .bashrc

Admin@Admin-PC ~
$ startxwin
startxwin:  Resource temporarily unavailable (errno 11):  Another X server insta
nce is running on DISPLAY :0

Admin@Admin-PC ~
$
```

```
Admin@Admin-PC ~
$ pwd
/home/Admin

Admin@Admin-PC ~
$ cd distvect/

Admin@Admin-PC ~/distvect
$ ns thro.tcl

Admin@Admin-PC ~/distvect
$ Cannot connect to existing nam instance. Starting a new one...
```



Auto layout: Ca 0.15 Cr 0.75 Iterations 10 ☑ Recalc

```
down
down
```

RESULT:

NAME: REG NO:

Ex.NO:10(b)	**SIMULATION OF LINK STATE ROUTING ALGORITHM**
DATE:	

AIM:

SOFTWARE REQUIRED:
NS-2

THEORY:
In **link state routing,** each router shares its knowledge of its neighborhood with every other router in the internet work. (i) **Knowledge about Neighborhood:** Instead of sending its entire routing table a router sends info about its neighborhood only. (ii) **To all Routers:** each router sends this information to every other router on the internet work not just to its neighbor .It does so by a process called **flooding. (iii)Information sharing when there is a change:** Each router sends out information about the neighbors when there is change.

PROCEDURE:
The Dijkstra algorithm follows four steps to discover what is called the **shortest path tree**(routing table) for each router:The algorithm begins to build the tree by identifying its roots. The root router's trees the router itself. The algorithm then attaches all nodes that can be reached from the root. The algorithm compares the tree's temporary arcs and identifies the arc with the lowest cumulative cost. This arc and the node to which it connects are now a permanent part of the shortest path tree. The algorithm examines the database and identifies every node that can be reached from its chosen node. These nodes and their arcs are added temporarily to the tree.
The last two steps are repeated until every node in the network has become a permanent part of the tree.

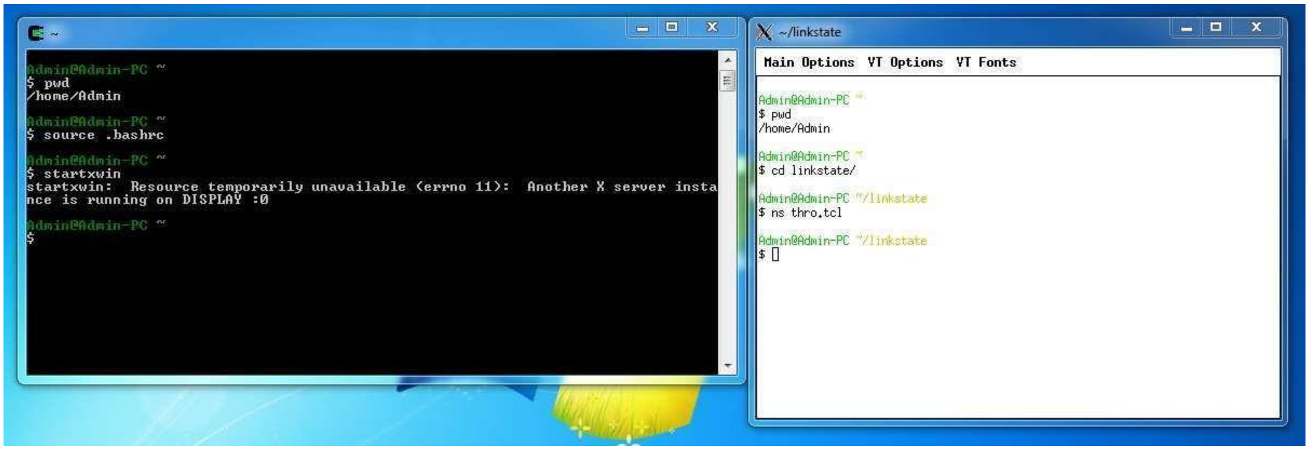
ALGORITHM:
1. Create a simulator object
2. Define different colors for different data flows
3. Open a nam trace file and define finish procedure then close the trace file, and execute nam on trace file.
4. Create n number of nodes using for loop
5. Create duplex links between the nodes
6. Setup UDP Connection between n(0) and n(5)
7. Setup another UDP connection between n(1) and n(5)
8. Apply CBR Traffic over both UDP connections
9. Choose Link state routing protocol to transmit data from sender to receiver.
10. Schedule events and run the program.

PROGRAM:

```
set ns [new Simulator]
set nr [open thro.tr w]
$ns trace-all $nr
set nf [open thro.nam w]
$ns namtrace-all $nf
proc finish { } {
global ns nr nf
$ns flush-trace
close $nf
close $nr
exec nam thro.nam &
exit 0
}

for { set i 0 } { $i < 12} { incr i 1 } {
set n($i) [$ns node]}
for {set i 0} {$i < 8} {incr i} {
$ns duplex-link $n($i) $n([expr $i+1]) 1Mb 10ms DropTail }
$ns duplex-link $n(0) $n(8) 1Mb 10ms DropTail
$ns duplex-link $n(1) $n(10) 1Mb 10ms DropTail
$ns duplex-link $n(0) $n(9) 1Mb 10ms DropTail
$ns duplex-link $n(9) $n(11) 1Mb 10ms DropTail
$ns duplex-link $n(10) $n(11) 1Mb 10ms DropTail
$ns duplex-link $n(11) $n(5) 1Mb 10ms DropTail

set udp0 [new Agent/UDP]
$ns attach-agent $n(0) $udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0
set null0 [new Agent/Null]
$ns attach-agent $n(5) $null0
$ns connect $udp0 $null0
set udp1 [new Agent/UDP]
$ns attach-agent $n(1) $udp1
set cbr1 [new Application/Traffic/CBR]
$cbr1 set packetSize_ 500
$cbr1 set interval_ 0.005
$cbr1 attach-agent $udp1
set null0 [new Agent/Null]
$ns attach-agent $n(5) $null0
$ns connect $udp1 $null0
$ns rtproto LS
$ns rtmodel-at 10.0 down $n(11) $n(5)
```

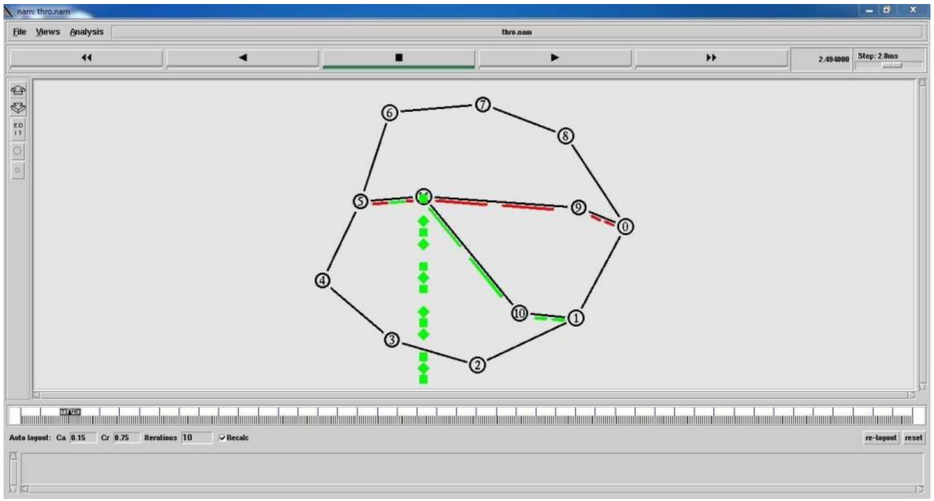
```
$ns rtmodel-at 15.0 down $n(7) $n(6)
$ns rtmodel-at 30.0 up $n(11) $n(5)
$ns rtmodel-at 20.0 up $n(7) $n(6)
$udp0 set fid_ 1
$udp1 set fid_ 2
$ns color 1 Red
$ns color 2 Green
$ns at 1.0 "$cbr0 start"
$ns at 2.0 "$cbr1 start"
$ns at 45 "finish"
$ns run
```

OUTPUT:



```
Admin@Admin-PC ~
$ pwd
/home/Admin

Admin@Admin-PC ~
$ source .bashrc

Admin@Admin-PC ~
$ startxwin
startxwin:  Resource temporarily unavailable (errno 11):  Another X server insta
nce is running on DISPLAY :0

Admin@Admin-PC ~
$
```

```
Admin@Admin-PC ~
$ pwd
/home/Admin

Admin@Admin-PC ~
$ cd linkstate/

Admin@Admin-PC ~/linkstate
$ ns thro.tcl

Admin@Admin-PC ~/linkstate
$ 
```



Ca 0.15 Cr 0.75 Iterations 10 ☑ Recalc

RESULT:

NAME: REG NO:

Ex.No:11	**Performance Evaluation of Routing protocols using Simulation tool.**
DATE:	

A) UNICAST PROTOCOL

AIM:

ALGORITHM:

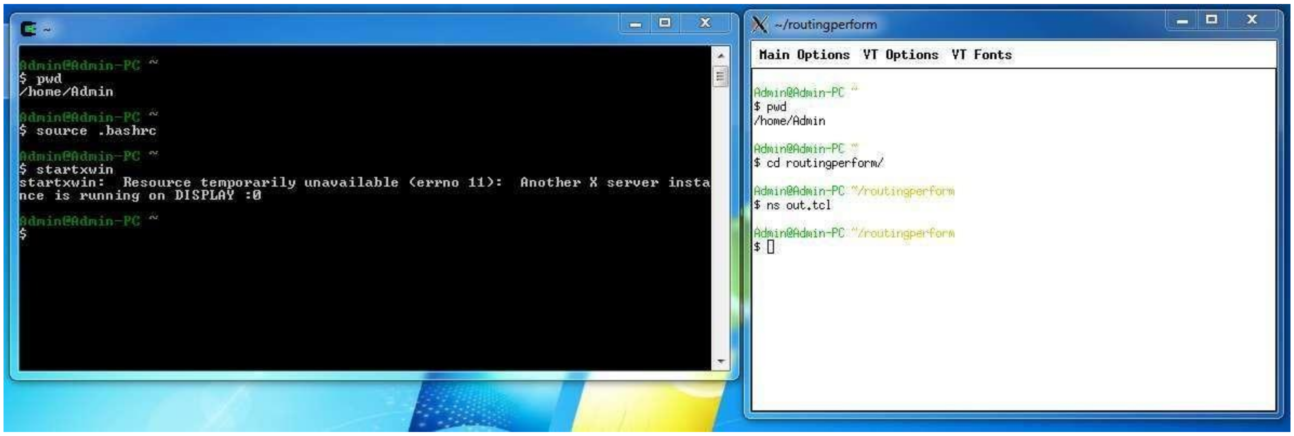
1. Start the program.
2. Declare the global variables ns for creating a new simulator.
3. Set the color for packets.
4. Open the network animator file in the name of file2 in the write mode.
5. Open the trace file in the name of file 1 in the write mode.
6. Set the unicast routing protocol to transfer the packets in network.
7. Create the required no of nodes.
8. Create the duplex-link between the nodes including the delay time,bandwidth and dropping queue mechanism.
9. Give the position for the links between the nodes.
10. Set a tcp reno connection for source node.
11. Set the destination node using tcp sink.
12. Setup a ftp connection over the tcp connection.
13. Down the connection between any nodes at a particular time.
14. Reconnect the downed connection at a particular time.
15. Define the finish procedure.
16. In the definition of the finish procedure declare the global variables ns, file1, and file2.
17. Close the trace file and name file and execute the network animation file.
18. At the particular time call the finish procedure.
19. Stop the program.

PROGRAM:

```
set ns [new Simulator]
#Define different colors for data flows (for NAM)
$ns color 1 Blue
$ns color 2 Red
#Open the Trace file
set file1 [open out.tr w]
$ns trace-all $file1
#Open the NAM trace file
set file2 [open out.nam w]
$ns namtrace-all $file2
#Define a 'finish' procedure
proc finish {}
{
global ns file1 file2
$ns flush-trace
close $file1
close $file2
exec nam out.nam &

exit 3
}
# Next line should be commented out to have the static routing
$ns rtproto DV
#Create six nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n4 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
#Create links between the nodes
$ns duplex-link $n0 $n1 0.3Mb 10ms DropTail
$ns duplex-link $n1 $n2 0.3Mb 10ms DropTail
$ns duplex-link $n2 $n3 0.3Mb 10ms DropTail
$ns duplex-link $n1 $n4 0.3Mb 10ms DropTail
$ns duplex-link $n3 $n5 0.5Mb 10ms DropTail
$ns duplex-link $n4 $n5 0.5Mb 10ms DropTail

#Give node position (for NAM)
$ns duplex-link-op $n0 $n1 orient right
$ns duplex-link-op $n1 $n2 orient right
$ns duplex-link-op $n2 $n3 orient up
$ns duplex-link-op $n1 $n4 orient up-left
$ns duplex-link-op $n3 $n5 orient left-up
$ns duplex-link-op $n4 $n5 orient right-up

#Setup a TCP connection
```

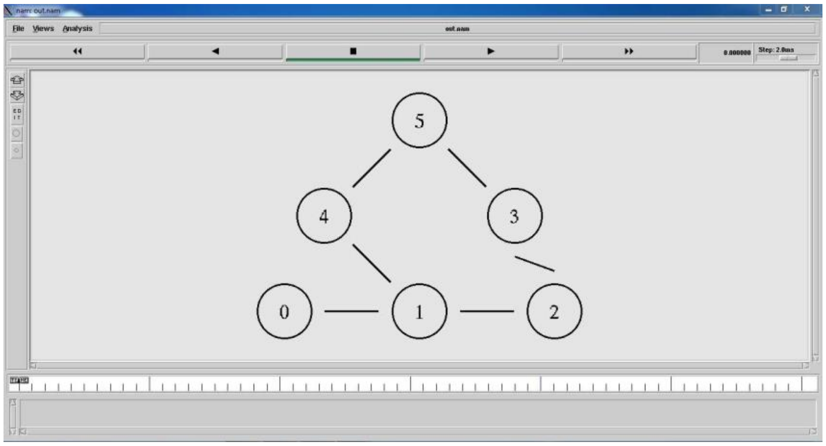
```
set tcp [new Agent/TCP/Newreno]
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink/DelAck]
$ns attach-agent $n5 $sink
$ns connect $tcp $sink
$tcp set fid_ 1
#Setup a FTP over TCP connection
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ftp set type_ FTP
$ns rtmodel-at 1.0 down $n1 $n4

$ns rtmodel-at 4.5 up $n1 $n4
$ns at 0.1 "$ftp start"
$ns at 6.0 "finish"
$ns run
```

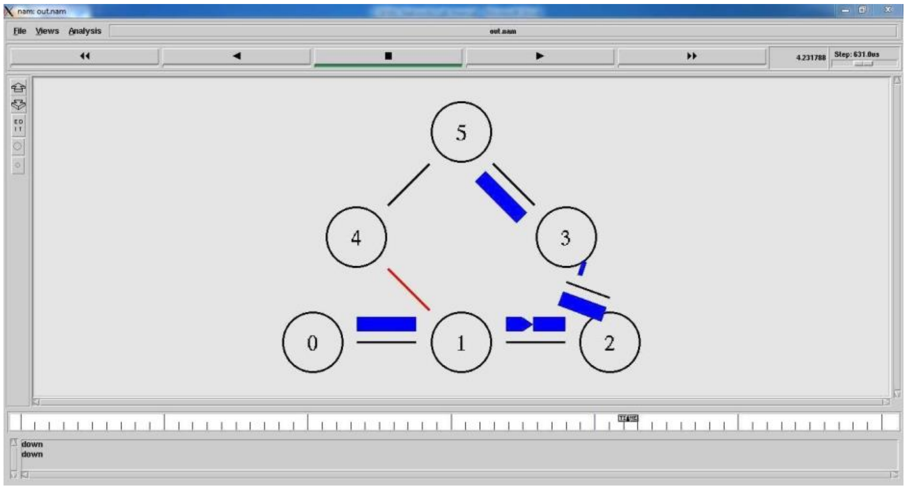
OUTPUT:



```
Admin@Admin-PC ~
$ pwd
/home/Admin

Admin@Admin-PC ~
$ source .bashrc

Admin@Admin-PC ~
$ startxwin
startxwin:  Resource temporarily unavailable (errno 11):  Another X server insta
nce is running on DISPLAY :0

Admin@Admin-PC ~
$
```

```
Admin@Admin-PC ~
$ pwd
/home/Admin

Admin@Admin-PC ~
$ cd routingperform/

Admin@Admin-PC ~/routingperform
$ ns out.tcl

Admin@Admin-PC ~/routingperform
$ 
```





```
down
down
```

(b) MULTICASTING ROUTING PROTOCOL

AIM:

ALGORITHM:

1. Start the program.
2. Declare the global variables ns for creating a new simulator.
3. Set the color for packets.
4. Open the network animator file in the name of file2 in the write mode.
5. Open the trace file in the name of file 1 in the write mode.
6. Set the multicast routing protocol to transfer the packets in network.
7. Create the multicast capable no of nodes.
8. Create the duplex-link between the nodes including the delay time,bandwidth and dropping queue mechanism.
9. Give the position for the links between the nodes.
10. Set a udp connection for source node.
11. Set the destination node ,port and random false for the source and destination files.
12. Setup a traffic generator CBR for the source and destination files.
13. Down the connection between any nodes at a particular time.
14. Create the receive agent for joining and leaving if the nodes in the group.
15. Define the finish procedure.
16. In the definition of the finish procedure declare the global variables.
17. Close the trace file and namefile and execute the network animation file.
18. At the particular time call the finish procedure.
19. Stop the program.

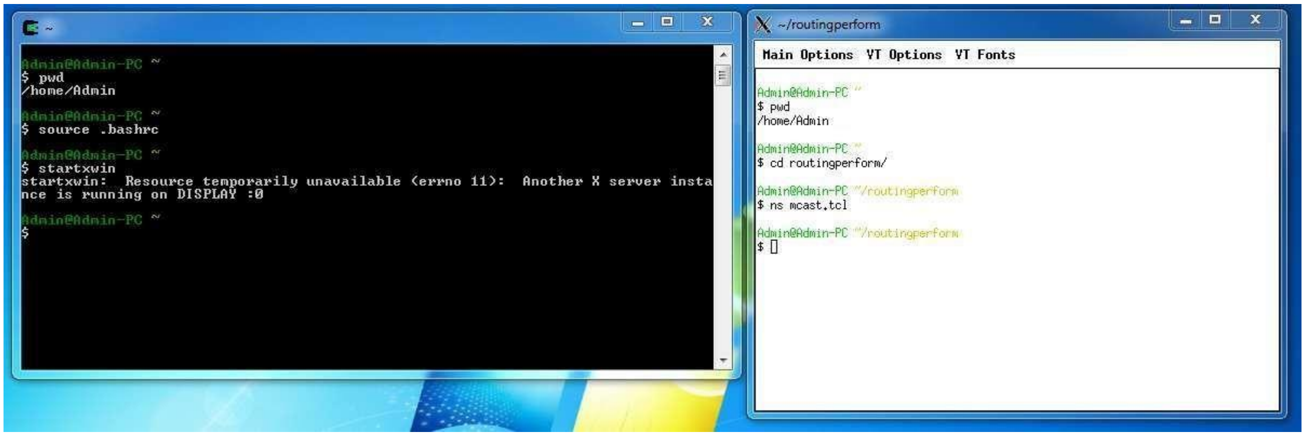
PROGRAM:

```
# Create scheduler
#Create an event scheduler wit multicast turned on
set ns [new Simulator -multicast on]
#$ns multicast
#Turn on Tracing
set tf [open output.tr w]
$ns trace-all $tf
# Turn on nam Tracing
set fd [open mcast.nam w]
$ns namtrace-all $fd
# Create nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
set n6 [$ns node]
set n7 [$ns node]
# Create links
$ns duplex-link $n0 $n2 1.5Mb 10ms DropTail
$ns duplex-link $n1 $n2 1.5Mb 10ms DropTail
$ns duplex-link $n2 $n3 1.5Mb 10ms DropTail
$ns duplex-link $n3 $n4 1.5Mb 10ms DropTail
$ns duplex-link $n3 $n7 1.5Mb 10ms DropTail
$ns duplex-link $n4 $n5 1.5Mb 10ms DropTail
$ns duplex-link $n4 $n6 1.5Mb 10ms DropTail
# Routing protocol: say distance vector
#Protocols: CtrMcast, DM, ST, BST
set mproto DM
set mrthandle [$ns mrtproto $mproto {}]
# Allocate group addresses
set group1 [Node allocaddr]
set group2 [Node allocaddr]
# UDP Transport agent for the traffic source
set udp0 [new Agent/UDP]

$ns attach-agent $n0 $udp0
$udp0 set dst_addr_ $group1
$udp0 set dst_port_ 0
set cbr1 [new Application/Traffic/CBR]
$cbr1 attach-agent $udp0
# Transport agent for the traffic source
set udp1 [new Agent/UDP]
$ns attach-agent $n1 $udp1
$udp1 set dst_addr_ $group2
```

```
$udp1 set dst_port_ 0
set cbr2 [new Application/Traffic/CBR]
$cbr2 attach-agent $udp1
# Create receiver
set rcvr1 [new Agent/Null]
$ns attach-agent $n5 $rcvr1
$ns at 1.0 "$n5 join-group $rcvr1 $group1"
set rcvr2 [new Agent/Null]
$ns attach-agent $n6 $rcvr2
$ns at 1.5 "$n6 join-group $rcvr2 $group1"
set rcvr3 [new Agent/Null]
$ns attach-agent $n7 $rcvr3
$ns at 2.0 "$n7 join-group $rcvr3 $group1"
set rcvr4 [new Agent/Null]
$ns attach-agent $n5 $rcvr1
$ns at 2.5 "$n5 join-group $rcvr4 $group2"
set rcvr5 [new Agent/Null]
$ns attach-agent $n6 $rcvr2
$ns at 3.0 "$n6 join-group $rcvr5 $group2"
set rcvr6 [new Agent/Null]
$ns attach-agent $n7 $rcvr3
$ns at 3.5 "$n7 join-group $rcvr6 $group2"
$ns at 4.0 "$n5 leave-group $rcvr1 $group1"
$ns at 4.5 "$n6 leave-group $rcvr2 $group1"
$ns at 5.0 "$n7 leave-group $rcvr3 $group1"
$ns at 5.5 "$n5 leave-group $rcvr4 $group2"
$ns at 6.0 "$n6 leave-group $rcvr5 $group2"
$ns at 6.5 "$n7 leave-group $rcvr6 $group2"
# Schedule events

$ns at 0.5 "$cbr1 start"
$ns at 9.5 "$cbr1 stop"
$ns at 0.5 "$cbr2 start"
$ns at 9.5 "$cbr2 stop"
#post-processing
$ns at 10.0 "finish"
proc finish {}
{
global ns tf
$ns flush-trace
close $tf
exec nam mcast.nam &
exit 0
}
# For nam
#Colors for packets from two mcast groups
$ns color 10 red
$ns color 11 green
```

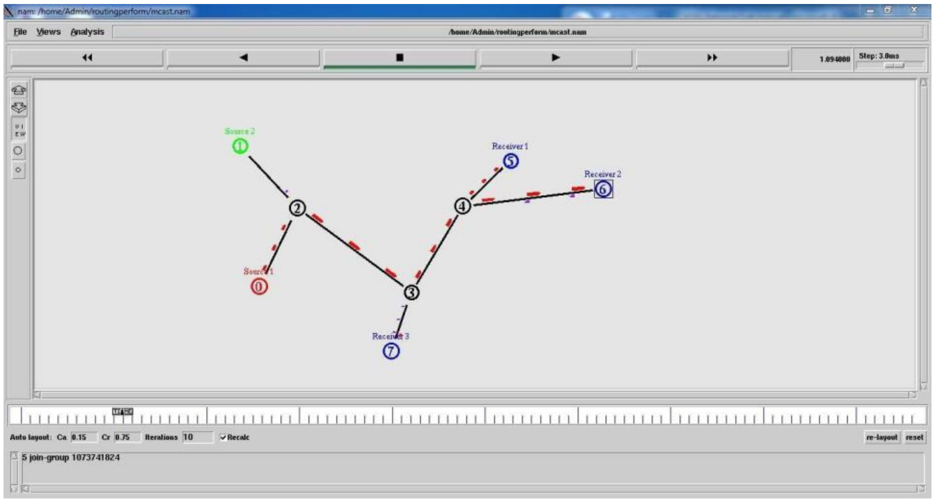
```
$ns color 30 purple
$ns color 31 green
# Manual layout: order of the link is significant!
#$ns duplex-link-op $n0 $n1 orient right
#$ns duplex-link-op $n0 $n2 orient right-up
#$ns duplex-link-op $n0 $n3 orient right-down
# Show queue on simplex link n0->n1
#$ns duplex-link-op $n2 $n3 queuePos 0.5
# Group 0 source
$udp0 set fid_ 10
$n0 color red
$n0 label "Source 1"
# Group 1 source
$udp1 set fid_ 11
$n1 color green
$n1 label "Source 2"
$n5 label "Receiver 1"
$n5 color blue
$n6 label "Receiver 2"
$n6 color blue
$n7 label "Receiver 3"
$n7 color blue
#$n2 add-mark m0 red
#$n2 delete-mark m0"
# Animation rate
$ns set-animation-rate 3.0ms
$ns run
```

OUTPUT:



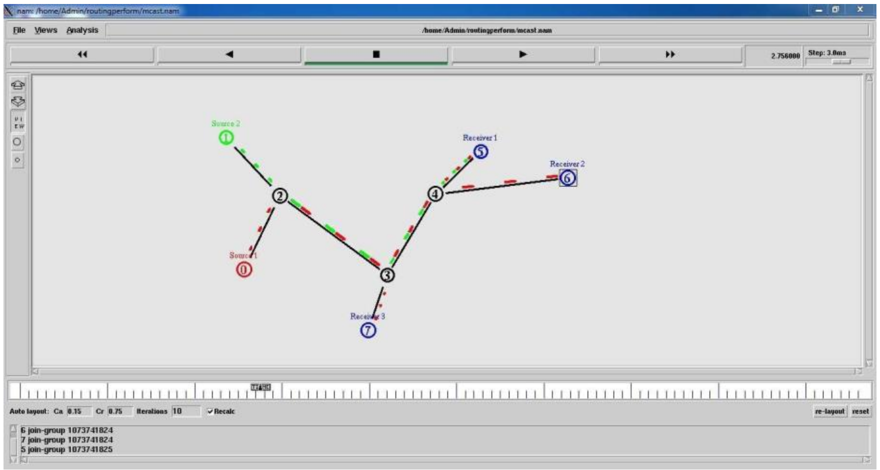
```
Admin@Admin-PC ~
$ pwd
/home/Admin

Admin@Admin-PC ~
$ source .bashrc

Admin@Admin-PC ~
$ startxwin
startxwin:  Resource temporarily unavailable (errno 11):  Another X server insta
nce is running on DISPLAY :0

Admin@Admin-PC ~
$
```

```
Admin@Admin-PC ~
$ pwd
/home/Admin

Admin@Admin-PC ~
$ cd routingperform/

Admin@Admin-PC ~/routingperform
$ ns mcast.tcl

Admin@Admin-PC ~/routingperform
$ 
```



Auto layout: Ca 0.15 Cr 0.75 Iterations 10 ✓Recalc

```
5 join-group 1073741824
```



Auto layout: Ca 0.15 Cr 0.75 Iterations 10 ✓Recalc

```
6 join-group 1073741824
7 join-group 1073741824
5 join-group 1073741825
```

NAME: REG NO:

RESULT:

NAME: REG NO:

Ex.No:12	**Simulation of ErrorDetection Code (like CRC)**
DATE:	

AIM:

ALGORITHM:

1. Start the Program
2. Given a bit string, append 0S to the end of it (the number of 0s is the same as the degree of the generator polynomial) let B(x) be the polynomial corresponding to B.
3. Divide B(x) by some agreed on polynomial G(x) (generator polynomial) and determine the remainder R(x). This division is to be done using Modulo 2 Division.
4. Define T(x) = B(x) –R(x)
5. (T(x)/G(x) => remainder 0)
6. Transmit T, the bit string corresponding to T(x).
7. Let T’ represent the bit stream the receiver gets and T’(x) the associated polynomial. The receiver divides T1(x) by G(x). If there is a 0 remainder, the receiver concludes T = T’ and no error occurred otherwise, the receiver concludes an error occurred and requires a retransmission
8. Stop the Program

NAME: REG NO:

PROGRAM:

```
import java.io.*;
class crc_gen
{
public static void main(String args[]) throws IOException {
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
int[] data;
int[] div;
int[] divisor;
int[] rem;
int[] crc;
int data_bits, divisor_bits, tot_length;
System.out.println("Enter number of data bits : "); data_bits=Integer.parseInt(br.readLine());
data=new int[data_bits];
System.out.println("Enter data bits : ");
for(int i=0; i<data_bits; i++)
data[i]=Integer.parseInt(br.readLine());
System.out.println("Enter number of bits in divisor : ");
divisor_bits=Integer.parseInt(br.readLine()); divisor=new int[divisor_bits];
System.out.println("Enter Divisor bits : ");
for(int i=0; i<divisor_bits; i++)
divisor[i]=Integer.parseInt(br.readLine());
System.out.print("Data bits are : ");
for(int i=0; i< data_bits; i++)
System.out.print(data[i]);
System.out.println();

System.out.print("divisor bits are : ");

for(int i=0; i< divisor_bits; i++)
System.out.print(divisor[i]);
System.out.println();
*/
tot_length=data_bits+divisor_bits-1;
div=new int[tot_length];
rem=new int[tot_length];
crc=new int[tot_length];
/*--------------------- CRC GENERATION---------------------------*/
for(int i=0;i<data.length;i++)
div[i]=data[i];
System.out.print("Dividend (after appending 0's) are : "); for(int i=0; i< div.length; i++)
System.out.print(div[i]);
System.out.println();
for(int j=0; j<div.length; j++){
rem[j] = div[j];
}
rem=divide(div, divisor, rem);
for(int i=0;i<div.length;i++)
{
//append dividend and remainder
crc[i]=(div[i]^rem[i]);
}
System.out.println();
System.out.println("CRC code : ");
for(int i=0;i<crc.length;i++)
System.out.print(crc[i]);
/*----------------------ERROR DETECTION------------------------*/
System.out.println();
System.out.println("Enter CRC code of "+tot_length+" bits : "); for(int i=0; i<crc.length; i++)
crc[i]=Integer.parseInt(br.readLine());
System.out.print("crc bits are : ");
for(int i=0; i< crc.length; i++)
System.out.print(crc[i]);
System.out.println();
for(int j=0; j<crc.length; j++){
rem[j] = crc[j];
}
rem=divide(crc, divisor, rem);
for(int i=0; i< rem.length; i++)
{
if(rem[i]!=0)
{

System.out.println("Error");
break;
}
if(i==rem.length-1)
System.out.println("No Error");
}
System.out.println("THANK  YOU...... )");
}
static int[] divide(int div[],int divisor[], int rem[])
{
int cur=0;
while(true)
{
for(int i=0;i<divisor.length;i++)
rem[cur+i]=(rem[cur+i]^divisor[i]);
while(rem[cur]==0 && cur!=rem.length-1)
cur++;
if((rem.length-cur)<divisor.length)
break;}return rem;}
```

OUTPUT :

```
Enter number of data bits :
7
Enter data bits :
1
0
1
1
0
0
1
Enter number of bits in divisor :
3
Enter Divisor bits :
1
0
1
Dividend (after appending 0's) are : 101100100
CRC code :
101100111
Enter CRC code of 9 bits :
1
0
1

1
0
0
1
0
1
crc bits are : 101100101
Error
THANK YOU ..... )
BUILD SUCCESSFUL (total time: 1 minute 34 seconds)
```

RESULT:

Ex.No:13	**Simulation of RIP and OSPF using NS2 /Qualnet/Jsim/OmneT++**
DATE:	

AIM:

THEORY:

Simulation is a very important technology in modern time. Computer assisted simulation can model hypothetical and real-life objects or activities on a computer to study the well-designed structure. A network simulator is a system of implementing the network on the computer through which the performance of the network is calculated. The computer assisted simulation technologies are applied in the simulation of networking algorithms. The functional network field is narrower than general simulation and it is natural that more specific requirements will be placed on network simulations.

Network simulator allows the researchers to test the scenarios that are difficult or expensive to simulate in real world. Design of various network topologies using nodes, hosts, hubs, bridges, routers and mobile units etc. is possible. The network simulators are of various types which can be compared on the basis of: range (simple to the complex), specification of nodes, links and traffic between the nodes. Specifying about the protocols used to handle traffic in a network, user friendly applications (allow users to easily visualize the simulated environment.), text-based applications (permit more advanced forms of customization) and programming-oriented tools (providing a programming framework that customizes to create an application that simulates the networking environment to be tested).

Network simulators are used by people from different areas such as academic researchers, industrialized sectors and Quality Assurance (QA) to design, simulate and analyze the performance of different network protocols. They can also be used to evaluate the outcome of the different parameters of the protocols being studied. Normally a network simulator comprises ofwide range of networking technologies and protocols that help users to build complex networks from basic building blocks like clusters of nodes and links. With their help, different network topologies can be designed using various types of nodes such as end-hosts, network bridges, routers, hubs, optical link-layer devices and mobile units. The following sections of the paper are organized as follows. Section 2, discusses about basic concepts in network simulation. Section 3, describes the various Simulators. In section 4, the preface of network simulators studied. In section 5, comparison of simulators is displayed and finally the conclusion in section 6.

Simulators

Most of the commercial simulators are Graphical User Interface (GUI) driven, while some network simulators are Command-Line Interface (CLI). The design of the network describes the state of the network (nodes, routers, switches and links) and the events (data transfer, transmission delay, packet error etc.). The major output of simulation is the trace files which log every packet and event that occurred during simulation and is used for analysis. Also provides other tools to facilitate visual analysis of trends and potential trouble spots. Most of the network simulators are discrete event, in which the list of pending "events" are stored and processed in order. Some events triggers the future events (i.e.) the event of the arrival of a packet at one node triggering the event of the arrival of that packet at a downstream node.

Simulation of networks is a very difficult task. For example, if blocking is high, then evaluation of the average occupancy is challenging because of high variance. To evaluate the probability of buffer overflow in a network, the time required for a precise answer can be enormously large. Techniques like "control variants" and “sampling" have been developed to speed simulation.

C++: C++ is fast to run but slower to change, making it suitable for detailed protocol implementation.

Otcl: OTcl runs much slower but can be changed very quickly (and interactively), making it ideal for simulation configuration. Ns provides glue to make objects and variables appear on both languages.

NS2 uses an OTcl interpreter by which the user writes an OTcl script that defines the network, (number of nodes and links) the transaction in the network (sources destinations, type of traffic) and the type of protocols used. The outcome of the simulation is a trace file that can be used for data processing (calculate delay, throughput etc). To visualize the simulation, a program called Network Animator (NAM) is used. It visualizes the packets as they propagate throughout the network. The ns- 2 simulator has numerous features that make it suitable for our simulations.

- A network environment for ad-hoc networks,
- Wireless channel modules (e.g.802.11),
- Routing along multiple paths,
- Mobile hosts for wireless cellular networks.
- Download of ns-2 source code is possible and can be compiled for multiple platforms.

OMNET++

It is a component-based, modular and open architecture discrete event simulator framework. The most common use of OMNeT++ is for simulation of networks, but it is also used for queuing network simulations and other areas as well. It is licensed under its own Academic Public License, which permits GNU Public License like freedom but only in noncommercial settings. It provides component architecture for models.

C++: The *C++ class library* comprises of simulation kernel and utility classes (for random number generation, statistics collection, topology discovery etc) -- this one is used to create simulation components (*simple modules* and channels); infrastructure to assemble simulations from these components and configure (*NED language, ini files*); runtime userinterfaces or *environments* for simulations (*Tkenv, Cmdenv*); an Eclipse-based simulation IDE for designing, running and evaluating simulations; extension interfaces for real-time simulation, emulation, MRIP, parallel distributed simulation, database connectivity and so on.

QualNet

It is a commercial network simulator from Scalable Network Technologies, Inc in 2000-2001. It is an ultra high fidelity network simulation software that predicts wired or wireless platform network and their device performance. For a large, heterogeneous network and distributed applications such networks are executed.

C++: For implementing new protocols, Qualnet uses C/C++ and follows a procedural paradigm. It uses the parallel simulationenvironment for the basic operations of complex systems (PARSEC). Hence it can run on distributed machines.

JSIM

J-Sim has been developed by a team at the Distributed Real-time Computing Laboratory (DRCL). The project has been sponsored by the National Science Foundation (NSF), DARPA's, Air Force Office of Scientific Research's Multidisciplinary University Research Initiative, the Ohio State University and University of Illinois at Urbana-Champaign. J-Sim is free and available with source code.

Java: Java is easy to learn and easy to use. In case of any problems, source texts provided with J-Sim can be used to create fresh code, compiled in the target environment, thus 100% compatible with JVM used. Java provides a class called Thread whose instances run parallel with other such instances. It is an object-oriented language, providing the conceptsof classes, instances, encapsulation, inheritance and polymorphism. J-Sim provides basic classes for simulation, process and queue which can be either directly used or extended according to specific user's requirements.

Tcl: Scripting is an essential part of J-Sim, used to "glue" all the components and define how the operation of the system takes place. It makes possible to manipulate Java objects in the Tcl environment, such as creating an object, invoking a method or accessing a field variable of a Java object.

Result:

TOPIC BEYOND SYLLABUS

Ex.no.14 a.	**Simulation of Go Back N protocol**
DATE:	

AIM:

ALGORITHM :

1. The source node transmits the frames continuously.
2. Each frame in the buffer has a sequence number starting from 1 and increasing up to the window size.
3. The source node has a window i.e. a buffer to store the frames. This buffer size is the number of frames to be transmitted continuously.

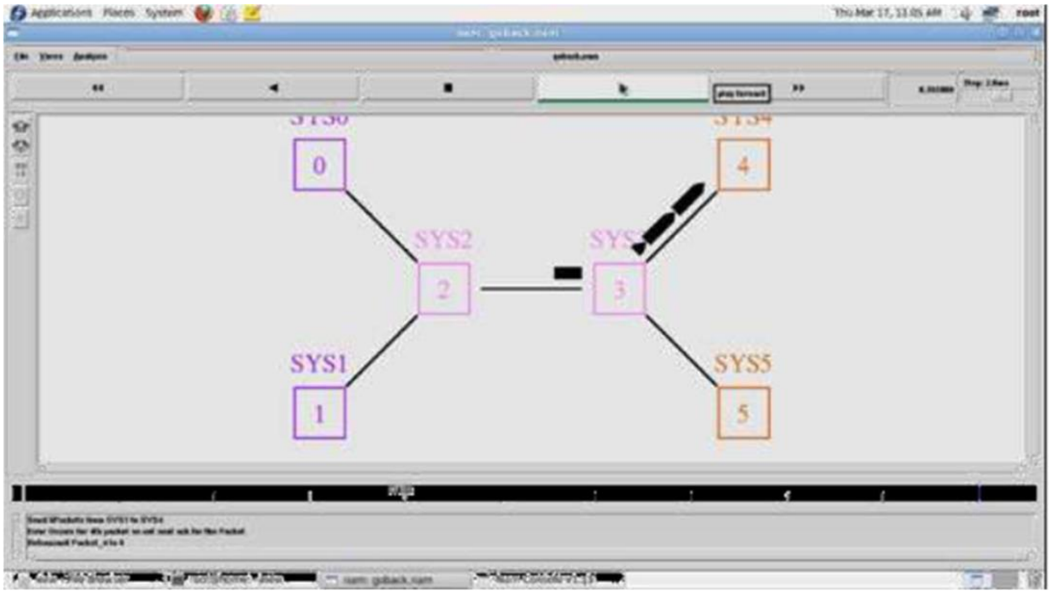
4. The size of the window depends on the protocol designer.
5. For the first frame, the receiving node forms a positive acknowledgement if the frame is received without error.
6. If subsequent frames are received without error (up to window size) cumulative positive acknowledgement is formed.
7. If the subsequent frame is received with error, the cumulative acknowledgment error-free frames are transmitted. If in the same window two frames or more frames are received with error, the second and the subsequent error frames are neglected. Similarly even the frames received without error after the receipt of a frame with error are neglected.
8. The source node retransmits all frames of window from the first error frame.
9. If the frames are errorless in the next transmission and if the acknowledgment is error free, the window slides by the number of error-free frames being transmitted.
10. If the acknowledgment is transmitted with error, all the frames of window at source are retransmitted, and window doesn’t slide.
11. This concept of repeating the transmission from the first error frame in the window is called as GOBACKN transmission flow control protocol.

PROGRAM :

```
#send packets one by one
set ns [new Simulator] set n0 [$ns node]
set n1 [$ns node] set n2 [$ns node] set n3 [$ns node] set n4 [$ns node] set n5 [$ns node] $n0
color "purple" $n1 color "purple" $n2 color "violet" $n3 color "violet" $n4 color "chocolate" $n5
color "chocolate" $n0 shape box ;
$n1 shape box ; $n2 shape box ; $n3 shape box ; $n4 shape box ; $n5 shape box ;
$ns at 0.0 "$n0 label SYS0"
$ns at 0.0 "$n1 label SYS1"
$ns at 0.0 "$n2 label SYS2"
$ns at 0.0 "$n3 label SYS3"
$ns at 0.0 "$n4 label SYS4"
$ns at 0.0 "$n5 label SYS5"
set nf [open goback.nam w] $ns namtrace-all $nf
set f [open goback.tr w] $ns trace-all $f
$ns duplex-link $n0 $n2 1Mb 20ms DropTail $ns duplex-link-op $n0 $n2 orient right-down $ns
queue-limit $n0 $n2 5
$ns duplex-link $n1 $n2 1Mb 20ms DropTail $ns duplex-link-op $n1 $n2 orient right-up $ns
duplex-link $n2 $n3 1Mb 20ms DropTail $ns duplex-link-op $n2 $n3 orient right
$ns duplex-link $n3 $n4 1Mb 20ms DropTail $ns duplex-link-op $n3 $n4 orient right-up $ns
duplex-link $n3 $n5 1Mb 20ms DropTail $ns duplex-link-op $n3 $n5 orient right-down
Agent/TCP set_nam_tracevar_true
set tcp [new Agent/TCP] $tcp set fid 1
$ns attach-agent $n1 $tcp
set sink [new Agent/TCPSink] $ns attach-agent $n4 $sink $ns connect $tcp $sink
set ftp [new Application/FTP] $ftp attach-agent $tcp
$ns at 0.05 "$ftp start"
$ns at 0.06 "$tcp set windowlnit 6"
$ns at 0.06 "$tcp set maxcwnd 6"

$ns at 0.25 "$ns queue-limit $n3 $n4 0"
$ns at 0.26 "$ns queue-limit $n3 $n4 10"
$ns at 0.305 "$tcp set windowlnit 4"
$ns at 0.305 "$tcp set maxcwnd 4"
$ns at 0.368 "$ns detach-agent $n1 $tcp ; $ns detach-agent $n4 $sink"
$ns at 1.5 "finish"
$ns at 0.0 "$ns trace-annotate \"Goback N end\""
$ns at 0.05 "$ns trace-annotate \"FTP starts at 0.01\""
$ns at 0.06 "$ns trace-annotate \"Send 6Packets from SYS1 to SYS4\""
$ns at 0.26 "$ns trace-annotate \"Error Occurs for 4th packet so not sent ack for the Packet\""
$ns at 0.30 "$ns trace-annotate \"Retransmit Packet_4 to 6\""
$ns at 1.0 "$ns trace-annotate \"FTP stops\""
proc finish {}
```

```
.

 {
global ns nf
$ns flush-trace
close $nf
puts "filtering..."
#exec tclsh../bin/namfilter.tcl goback.nam
#puts "running nam..."
exec nam goback.nam &
exit 0
}
$ns run.
```

OUTPUT



```
Send 6Packets from SYS1 to SYS4
Error Occurs for 4th packet so not sent ack for the Packet
Retransmit Packet_4 to 6
```

RESULT:

NAME: REG NO:

EX.NO:14 b	**CARRIER SENSE MULTIPLE ACCESS**
DATE:	

AIM:

ALGORITHM:

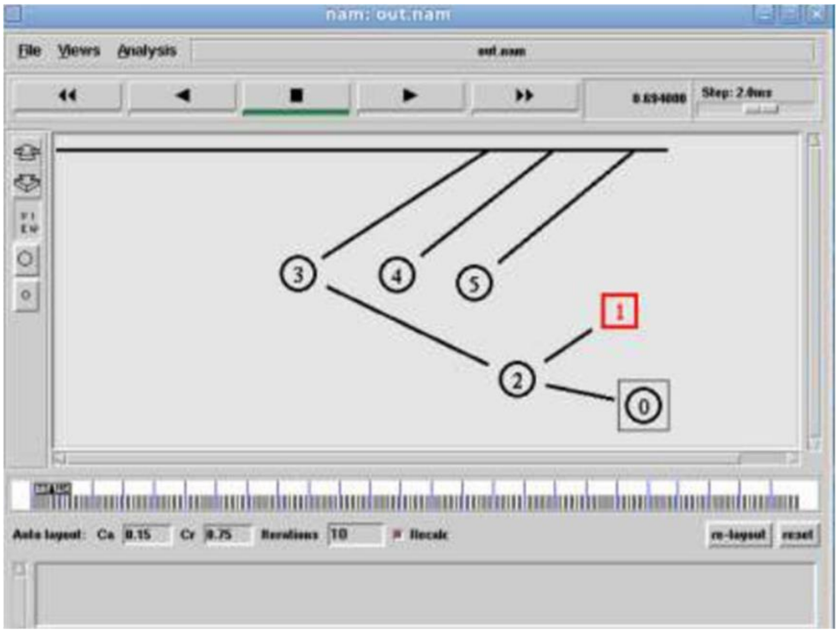
1. Start the program.
2. Declare the global variables ns for creating a new simulator.
3. Set the color for packets.
4. Open the network animator file in the write mode.
5. Open the trace file and the win file in the write mode.
6. Transfer the packets in network.
7. Create the capable no of nodes.
8. Create the duplex-link between the nodes including the delay time,bandwidth and dropping queue mechanism.
9. Give the position for the links between the nodes.
10. Set a tcp connection for source node.
11. Set the destination node using tcp sink.
12. Set the window size and the packet size for the tcp.
13. Set up the ftp over the tcp connection.
14. Set the udp and tcp connection for the source and destination.
15. Create the traffic generator CBR for the source and destination files.
16. Define the plot window and finish procedure.
17. In the definition of the finish procedure declare the global variables.
18. Close the trace file and namefile and execute the network animation file.
19. At the particular time call the finish procedure.
20. Stop the program.

PROGRAM:

```
set ns [new Simulator]
$ns color 1 blue
$ns color 2 red
set fi1 [open out.tr w]
set winfile [open WinFile w]
$ns trace-all $fi1
set fi2 [open out.nam w]
$ns namtrace-all $fi2
proc finish {}
{
global ns fi1 fi2
$ns flush-trace
close $fi1
close $fi2
exec nam out.nam &
exit 0
}
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]

set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
$n1 color red
$n1 shape box
$ns duplex-link $n0 $n2 2Mb 10ms DropTail
$ns duplex-link $n1 $n2 2Mb 10ms DropTail
$ns simplex-link $n2 $n3 0.3Mb 100ms DropTail
$ns simplex-link $n3 $n2 0.3Mb 100ms DropTail
set lan [$ns newLan "$n3 $n4 $n5" 0.5Mb 40ms LL Queue/DropTail MAC/Csma/Cd Channel]
set tcp [new Agent/TCP/Newreno]
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink/DelAck]
$ns attach-agent $n4 $sink
$ns connect $tcp $sink
$tcp set fid_ 1
$tcp set window_ 8000
$tcp set packetsize_ 552
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ftp set type_ FTP
set udp [new Agent/UDP]
$ns attach-agent $n1 $udp
```

```
set null [new Agent/Null]
$ns attach-agent $n5 $null
$ns connect $udp $null
$udp set fid_ 2
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set type_ CBR
$cbr set packet_size_ 1000
$cbr set rate_ 0.01mb
$cbr set random_ false
$ns at 0.1 "$cbr start"
$ns at 1.0 "$ftp start"
$ns at 24.0 "$ftp stop"
$ns at 24.5 "$cbr stop"
proc plotwindow { tcpSource file }
{
global ns
set time 0.1

set now [$ns now]
set cwnd [$tcpSource set cwnd_]
set wnd [$tcpSource set window_]
puts $file "$now $cwnd"
$ns at [expr $now+$time] "plotwindow $tcpSource $file"
}
$ns at 1.0 "plotwindow $tcp $winfile"
$ns at 5 "$ns trace-annotate \"packet drop\""
$ns at 125.0 "finish"
$ns run
```

OUTPUT:

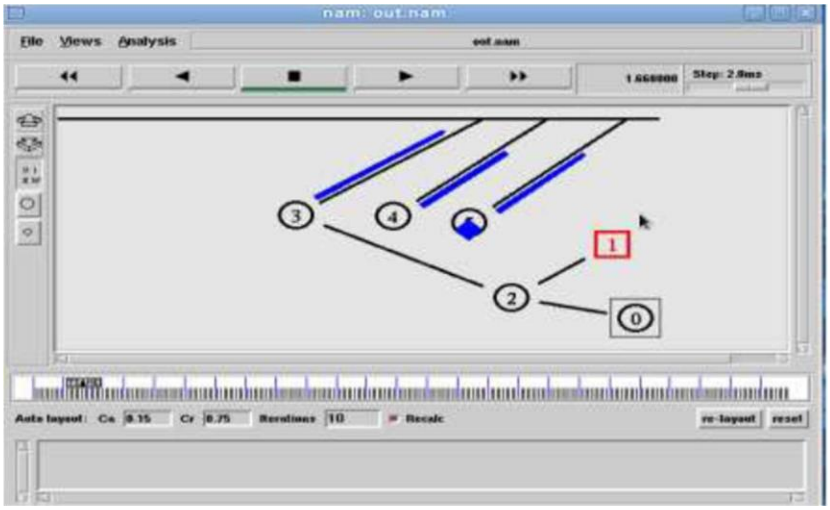


nam: out.nam

out.nam

0.694000 Step: 2.0ms

Auto layout: Ca 0.15 Cr 0.75 Iterations 10 Recalc



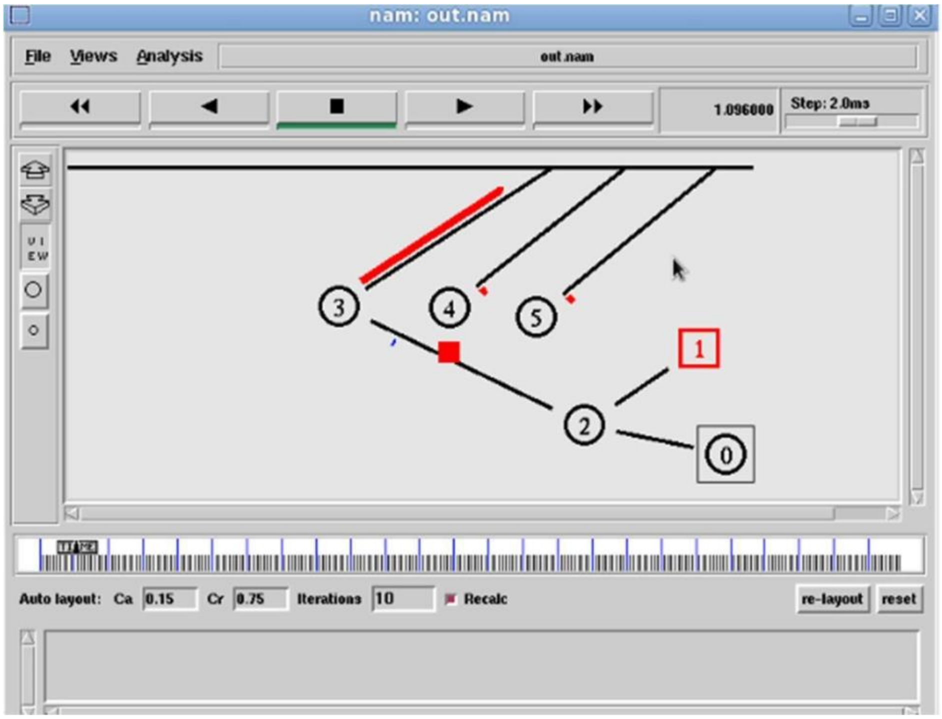
nam: out.nam

out.nam

1.666000 Step: 2.0ms

Auto layout: Ca 0.15 Cr 0.75 Iterations 10 Recalc

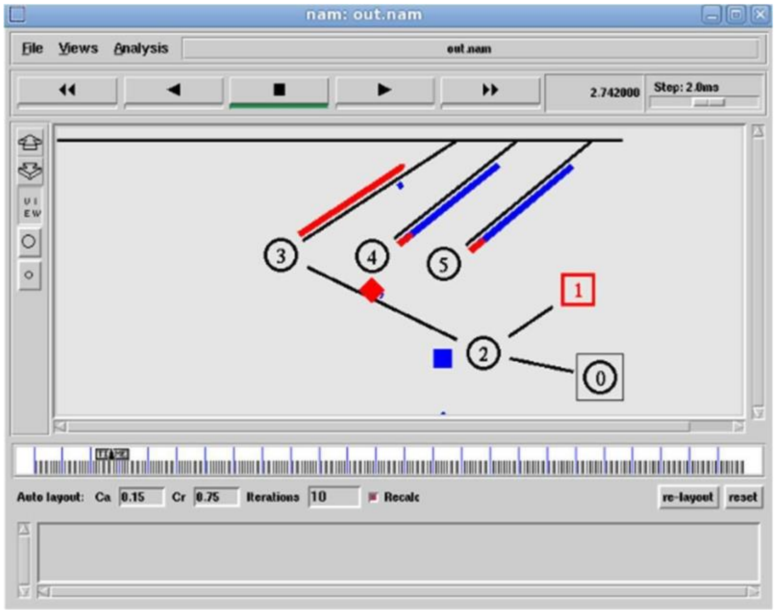
NAME: REG NO:



1.096000

Step: 2.0ms

Auto layout: Ca 0.15 Cr 0.75 Iterations 10 Recalc



2.742000

Step: 2.0ms

Auto layout: Ca 0.15 Cr 0.75 Iterations 10 Recalc

RESULT

NAME: REG NO: